```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE PIPMOD
OBJECT MODULE PLACED IN PIP.OBJ
COMPILER INVOKED BY:  PLM80 PIP.PLM DEBUG OPTIMIZE


           $title('PERIPHERAL INTERCHANGE PROGRAM')
  1        PIPMOD:
              DO;
           /* P E R I P H E R A L   I N T E R C H A N G E   P R O G R A M

                 COPYRIGHT (C) 1976, 1977, 1978, 1979, 1980, 1981, 1982
                 DIGITAL RESEARCH
                 BOX 579
                 PACIFIC GROVE, CA
                 93950

                 Revised:
                    17 Jan 80  by  Thomas Rolander (MP/M 1.1)
                    05 Oct 81  by  Ray Pedrizetti  (MP/M-86 2.0)
                    18 Dec 81  by  Ray Pedrizetti  (CP/M-86 1.1)
                    29 Jun 82  by  Ray Pedrizetti  (CCP/M-86 3.0) */

           /*    Command lines used for CMD file generation        */

           /* (on VAX)
              asm86 scd1.a86
                 asm86 inpout.a86
              plm86 pip.plm debug xref optimize(3)
              link86 scd1.obj,inpout.obj,pip.obj, to pip.lnk
              loc86 pip.lnk od(sm(code,dats,data,const,stack)) -
                       ad(sm(code(0), dats(10000h)))  ss(stack(+32)) to pip.
              h86 pip

             (on a micro)
              vax pip.h86 $fans
              gencmd pip data[b1000 m280 xfff]

                 * note the beginning of the data segment will change when
              * the program is changed. see the 'MP2' file generated
              * by LOC86.  the constants are last to force hex generation
             */

             /* Compiler Directives */
           /** $set (mpm) **/
           /** $reset (cpm3) **/
           /** $cond **/

  2   1    declare    /* resets stack for error handling */
              reset label external;

  3   1    DECLARE
              MAXB  ADDRESS EXTERNAL,     /* ADDR FIELD OF JMP BDOS */
              FCB (33) BYTE EXTERNAL,     /* DEFAULT FILE CONTROL BLOCK */
              BUFF(128)BYTE EXTERNAL;     /* DEFAULT BUFFER */

  4   1    declare
```

CP/M plus Ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CPS-135

```
                retry byte initial(0);  /* true if error has occured */

 5   1      OUTD: PROCEDURE(B) external;
 6   2          DECLARE B BYTE;
                /* SEND B TO OUT: DEVICE */
 7   2          END OUTD;

 8   1      INPD: PROCEDURE BYTE external;
 9   2          END INPD;

10   1      MON1: PROCEDURE(F,A) EXTERNAL;
11   2          DECLARE F BYTE,
                A ADDRESS;
12   2          END MON1;

13   1      MON2: PROCEDURE(F,A) BYTE EXTERNAL;
14   2          DECLARE F BYTE,
                A ADDRESS;
15   2          END MON2;

16   1      MON3: PROCEDURE(F,A) ADDRESS EXTERNAL;
17   2          DECLARE F BYTE,
                A ADDRESS;
18   2          END MON3;


19   1      plm: procedure public;

20   2      DECLARE
            /** $if mpm **/
               VERSION  LITERALLY '0031H', /* REQUIRED FOR BDOS 3.1 OPERATION */
            /** $else **/
            /** $endif **/

               ENDFILE LITERALLY '1AH';    /* END OF FILE MARK */

21   2      DECLARE COPYRIGHT(*) BYTE DATA (
            /** $if cpm3 **/
                   ' (12/06/82) CP/M 3 PIP VERS 3.0 ');
            /** $else **/
            /** $endif **/


               /* LITERAL DECLARATIONS */
22   2      DECLARE
               LIT  LITERALLY 'LITERALLY',
               LPP  LIT '60',     /* LINES PER PAGE */
               TAB  LIT '09H',    /* HORIZONTAL TAB */
               FF   LIT '0CH',    /* FORM FEED */
               LA   LIT '05FH',   /* LEFT ARROW */
               LB   LIT '05BH',   /* LEFT BRACKET */
               RB   LIT '05DH',   /* RIGHT BRACKET */

               FSIZE  LIT '33',
               FRSIZE LIT '36',   /* SIZE OF RANDOM FCB */
               NSIZE  LIT '8',
               FNSIZE LIT '11',
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
            FEXT  LIT '9',
            FEXTL LIT '3',

               /* scanner return type code */
            outt     LIT '0',   /* output device */
            PRNT     LIT '1',   /* PRINTER */
            LSTT     LIT '2',   /* list device */
            axot     lit '3',   /* auxilary output device */
            FILE     LIT '4',   /* file type */
            axit     lit '5',   /* auxilary input/output device */
            CONS     LIT '6',   /* CONSOLE */
            axit     LIT '7',   /* auxilary input device */
            inpt     lit '8',   /* input device */
            NULT     LIT '9',   /* nul characters */
            EOFT     LIT '10',   /* EOF character */
            ERR      LIT '11',  /* error type */
            SPECL    LIT '12',  /* special character */
            DISKNAME LIT '13';  /* diskname letter */

23  2       DECLARE
               SEARFCB LIT 'FCB';   /* SEARCH FCB IN MULTI COPY */

24  2       DECLARE
               TRUE    LIT '1',
               FALSE   LIT '0',
               FOREVER LIT 'WHILE TRUE',
               cntrlc  lit '3',
               CR      LIT '13',
               LF      LIT '10',
               WHAT    LIT '63';

            /** $if mpm **/
25  2       declare
               maxmcnt lit '128',    /* maximum multi sector count */
               maxmbuf lit '16384';  /* maximum multi sector buffer size */
            /** $endif **/

26  2       DECLARE
               COLUMN   BYTE,      /* COLUMN COUNT FOR PRINTER TABS */
               LINENO   BYTE,      /* LINE WITHIN PAGE */
               FEEDBASE BYTE,      /* USED TO FEED SEARCH CHARACTERS */
               FEEDLEN  BYTE,      /* LENGTH OF FEED STRING */
               MATCHLEN BYTE,      /* USED IN MATCHING STRINGS */
               QUITLEN  BYTE,      /* USED TO TERMINATE QUIT COMMAND */
               CDISK    BYTE,      /* CURRENT DISK */
               SBLEN    ADDRESS,   /* SOURCE BUFFER LENGTH */
               DBLEN    ADDRESS,   /* DEST BUFFER LENGTH */
               tblen    address,   /* temp buffer length */
               SBASE    ADDRESS,   /* SOURCE BUFFER BASE */

               /* THE VECTORS DBUFF AND SBUFF ARE DECLARED WITH DIMENSION
               1024, BUT ACTUALLY VARY WITH THE FREE MEMORY SIZE */
               DBUFF(1024) BYTE AT (.MEMORY), /* DESTINATION BUFFER */
               SBUFF BASED SBASE (1024) BYTE, /* SOURCE BUFFER */

                  /* source fcb, password and password mode */
               source structure (
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
        fcb(frsize)  byte,
/** $if mpm **/
        pwnam(nsize) byte,
        pwmode       byte,
/** $endif **/
        user         byte,
        type         byte ),

        /* temporary destination fcb, password and password mode */
    dest structure (
        fcb(frsize)  byte,
/** $if mpm **/
        pwnam(nsize) byte,
        pwmode       byte,
/** $endif **/
        user         byte,
        type         byte ),

        /* original destination fcb, password and password mode */
    odest structure (
        fcb(frsize)  byte,
/** $if mpm **/
        pwnam(nsize) byte,
        pwmode       byte,
/** $endif **/
        user         byte,
        type         byte ),

    filsize(3) byte,     /* file size random record number */

    DESTR    ADDRESS AT(.DEST.FCB(33)),    /* RANDOM RECORD POSITION */
    SOURCER  ADDRESS AT(.SOURCE.FCB(33)),  /* RANDOM RECORD POSITION */
    DESTR2   BYTE    AT(.DEST.FCB(35)),    /* RANDOM RECORD POSITION R2 */
    SOURCER2 BYTE    AT(.SOURCE.FCB(35)),  /* RANDOM RECORD POSITION R2 */

    extsave  byte,      /* temp extent byte for bdos bug */

    nsbuf    address,   /* next source buffer */
/** $if mpm **/
    bufsize address,    /* multsect buffer size */
    nseccnt  byte,      /* last multi sector count value */
/** $endif **/
    NSOURCE ADDRESS,    /* NEXT SOURCE CHARACTER */
    NDEST   ADDRESS;    /* NEXT DESTINATION CHARACTER */

27  2   DECLARE
        fastcopy byte,      /* true if copy directly to dbuf */
        dblbuf   byte,      /* true if both source and dest buffer used */
        concat   byte,      /* true if concatination command */
        ambig    byte,      /* true if file is ambig type */
        dfile    byte,      /* true if dest is file type */
        sfile    byte,      /* true if source is file type */
        made     byte,      /* true if destination file already made */
        opened   byte,      /* true if source file open */
        endofsrc byte,      /* true if end of source file */
        nendcmd  byte,      /* true if not end of command tail */
        insparc  byte,      /* true if in middle of sparce file */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
            sparfil  byte,      /* true if sparce file being copied */
            MULTCOM  BYTE,      /* true if processing multiple commands */
            PUTNUM   BYTE,      /* SET WHEN READY FOR NEXT LINE NUM */
            CONCNT   BYTE,      /* COUNTER FOR CONSOLE READY CHECK */
            CHAR     BYTE,      /* LAST CHARACTER SCANNED */
            FLEN     BYTE;      /* FILE NAME LENGTH */

 28  2      declare
            f1      byte,       /* f1 user attribute flag */
            f2      byte,       /* f2 user attribute flag */
            f3      byte,       /* f3 user attribute flag */
            f4      byte,       /* f4 user attribute flag */
            ro      byte,       /* read only attribute flag */
            sys     byte,       /* system attribute flag */
          /** $if mpm **/
            exten   byte,       /* extention error code */
            odcnt   byte,       /* saves dcnt for open dest file */
            eretry  byte,       /* error return flag */
          /** $endif **/
            dcnt    byte;       /* error code or directory code */

 29  2      DECLARE CBUFF(130) BYTE,   /* COMMAND BUFFER */
            MAXLEN       BYTE AT (.CBUFF(0)),  /* MAX BUFFER LENGTH */
            COMLEN       BYTE AT (.CBUFF(1)),  /* CURRENT LENGTH */
            COMBUFF(128) BYTE AT (.CBUFF(2)),  /* COMMAND BUFFER CONTENTS */
            CBP          BYTE;                 /* COMMAND BUFFER POINTER */

 30  2      DECLARE
            CUSER BYTE, /* CURRENT USER NUMBER */
            last$user byte;

 31  2      DECLARE /* CONTROL TOGGLE VECTOR */
            CONT(26) BYTE,   /* ONE FOR EACH ALPHABETIC */
            /* 00 01 02 03 04 05 06 07 08 09 10 11 12 13
                  A  B  C  D  E  F  G  H  I  J  K  L  M  N
               14 15 16 17 18 19 20 21 22 23 24 25
                  O  P  Q  R  S  T  U  V  W  X  Y  Z   */
            archiv byte  at(.cont(0)),    /* file archive */
            confrm byte  at(.cont(2)),    /* confirm copy */
            DELET  BYTE  AT(.CONT(3)),    /* DELETE CHARACTERS */
            ECHO   BYTE  AT(.CONT(4)),    /* ECHO CONSOLE CHARACTERS */
            FORMF  BYTE  AT(.CONT(5)),    /* FORM FILTER */
            GETU   BYTE  AT(.CONT(6)),    /* GET FILE, USER # */
            HEXT   BYTE  AT(.CONT(7)),    /* HEX FILE TRANSFER */
            IGNOR  BYTE  AT(.CONT(8)),    /* IGNORE :00 RECORD ON FILE */
            Kilds  byte  at(.cont(10)),   /* kill filename display */
            LOWER  BYTE  AT(.CONT(11)),   /* TRANSLATE TO LOWER CASE */
            NUMB   BYTE  AT(.CONT(13)),   /* NUMBER OUTPUT LINES */
            OBJ    BYTE  AT(.CONT(14)),   /* OBJECT FILE TRANSFER */
            PAGCNT BYTE  AT(.CONT(15)),   /* PAGE LENGTH */
            QUITS  BYTE  AT(.CONT(16)),   /* QUIT COPY */
            RSYS   BYTE  AT(.CONT(17)),   /* READ SYSTEM FILES */
            STARTS BYTE  AT(.CONT(18)),   /* START COPY */
            TABS   BYTE  AT(.CONT(19)),   /* TAB SET */
            UPPER  BYTE  AT(.CONT(20)),   /* UPPER CASE TRANSLATE */
            VERIF  BYTE  AT(.CONT(21)),   /* VERIFY EQUAL FILES ONLY */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                WRROF  BYTE  AT(.CONT(22)),    /* WRITE TO R/O FILE */
                ZEROP  BYTE  AT(.CONT(25));    /* ZERO PARITY ON INPUT */

 32   2     DECLARE ZEROSUP BYTE,  /* ZERO SUPPRESSION */
                (C3,C2,C1) BYTE;     /* LINE COUNT ON PRINTER  */


             /** $if mpm **/
 33   2     retcodes: procedure(a);
 34   3        declare a address;
 35   3        dcnt = low(a);
 36   3        exten = high(a);
 37   3        end retcodes;
             /** $endif **/

 38   2     BOOT: PROCEDURE;
                /* SYSTEM REBOOT */
 39   3        CALL MON1(0,0);
 40   3        END BOOT;


 41   2     RDCHAR: PROCEDURE BYTE;
                /* READ CONSOLE CHARACTER */
 42   3        RETURN MON2(1,0);
 43   3        END RDCHAR;

 44   2     PRINTCHAR: PROCEDURE(CHAR);
 45   3        DECLARE CHAR BYTE;
 46   3        CALL MON1(2,CHAR AND 7FH);
 47   3        END PRINTCHAR;

 48   2     CRLF: PROCEDURE;
 49   3        CALL PRINTCHAR(CR);
 50   3        CALL PRINTCHAR(LF);
 51   3        END CRLF;

 52   2     printx: procedure(a);
 53   3        declare a address;
 54   3        call mon1(9,a);
 55   3        end printx;

 56   2     PRINT: PROCEDURE(A);
 57   3        DECLARE A ADDRESS;
                /* PRINT THE STRING STARTING AT ADDRESS A UNTIL THE
                NEXT DOLLAR SIGN IS ENCOUNTERED */
 58   3        CALL CRLF;
 59   3        CALL printx(A);
 60   3        END PRINT;

 61   2     RDCOM: PROCEDURE;
                /* READ INTO COMMAND BUFFER */
 62   3        MAXLEN = 128;
 63   3        CALL MON1(10,.MAXLEN);
 64   3        END RDCOM;

 65   2     CVERSION: PROCEDURE ADDRESS;
 66   3        RETURN MON3(12,0); /* VERSION NUMBER */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 67  3          END CVERSION;

 68  2      SETDMA: PROCEDURE(A);
 69  3          DECLARE A ADDRESS;
 70  3          CALL MON1(26,A);
 71  3          END SETDMA;

             /** $if mpm **/
 72  2      setpw: procedure(fcba);
 73  3          declare fcba address;
 74  3          declare fcbs based fcba structure (
                     fcb(frsize) byte,
                     pwnam(nsize) byte );
 75  3          call setdma(.fcbs.pwnam(0));
 76  3          end setpw;
             /** $endif **/

 77  2      OPEN: PROCEDURE(fcba);
 78  3          DECLARE fcba ADDRESS;
 79  3          declare fcb based fcba (frsize) byte;
             /** $if mpm **/
 80  3          CALL SETPW(fcba);
 81  3          call retcodes(mon3(15,fcba));
             /** $else **/
             /** $endif **/
 82  3          if dcnt <> 255 and rol(fcb(8),1) then
 83  3              do; call mon1(16,fcba);
 85  4              dcnt = 255;
             /** $if mpm **/
 86  4              exten = 0;
             /** $endif **/
 87  4              end;
 88  3          END OPEN;

 89  2      CLOSE: PROCEDURE(FCB);
 90  3          DECLARE FCB ADDRESS;
             /** $if mpm **/
 91  3          call retcodes(MON3(16,FCB));
             /** $else **/
             /** $endif **/
 92  3          END CLOSE;

 93  2      SEARCH: PROCEDURE(FCB);
 94  3          DECLARE FCB ADDRESS;
             /** $if mpm **/
 95  3          call retcodes(MON3(17,FCB));
             /** $else **/
             /** $endif **/
 96  3          END SEARCH;

 97  2      SEARCHN: PROCEDURE;
             /** $if mpm **/
 98  3          call retcodes(MON3(18,0));
             /** $else **/
             /** $endif **/
 99  3          END SEARCHN;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
100   2     DELETE: PROCEDURE(FCB);
101   3        DECLARE FCB ADDRESS;
           /** $if mpm **/
102   3        CALL SETPW(FCB);
103   3        call retcodes(MON3(19,FCB));
           /** $else **/
           /** $endif **/
104   3        END DELETE;

105   2     DISKRD  PROCEDURE(FCB);
106   3        DECLARE FCB ADDRESS;
           /** $if mpm **/
107   3        call retcodes(MON3(20,FCB));
           /** $else **/
           /** $endif **/
108   3        ENT DISKRD;

109   2     DISKWRITE: PROCEDURE(FCB);
110   3        DECLARE FCB ADDRESS;
           /** $if mpm **/
111   3        call retcodes(MON3(21,FCB));
           /** $else **/
           /** $endif **/
112   3        END DISKWRITE;

113   2     MAKE: procedure(fcba);
114   3        declare fcba address;
           /** $if mpm **/
115   3        declare fcbs based fcba structure (
                    fcb(frsize) byte,
                    pwnam(nsize) byte );
116   3        if fcbs.pwnam(0) = 0 then   /* zero if no password */
117   3           fcbs.fcb(6) = fcbs.fcb(6) and 7fh;  /* reset password attribute */
118   3        else do;
119   4           fcbs.fcb(6) = fcbs.fcb(6) or 80h;  /* set password attribute */
120   4           call setdma(.fcbs.pwnam(0));  /* set password dma */
121   4           end;
122   3        call retcodes(mon3(22,fcba));
           /** $else **/
           /** $endif **/
123   3        END MAKE;

124   2     RENAME: PROCEDURE(FCB);
125   3        DECLARE FCB ADDRESS;
           /** $if mpm **/
126   3        CALL SETPW(FCB);
127   3        call retcodes(MON3(23,FCB)) ;
           /** $else **/
           /** $endif **/
128   3        END RENAME;

129   2     getdisk: procedure byte;
130   3        return mon2(25,0);
131   3        end getdisk;

132   2     SETIND: PROCEDURE(FCB);
133   3        DECLARE FCB ADDRESS;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                /** $if mpm **/
 134   3            call retcodes(MON3(30,FCB));
                /** $else **/
                /** $endif **/
 135   3            END SETIND;

 136   2      GETUSER: PROCEDURE BYTE;
 137   3            RETURN MON2(32,0FFH);
 138   3            END GETUSER;

 139   2      SETUSER: PROCEDURE(USER);
 140   3            DECLARE USER BYTE;
 141   3            if last$user <> user then
 142   3               CALL MON1(32,(last$user:=USER));
 143   3            END SETUSER;

 144   2      SETCUSER: PROCEDURE;
 145   3            CALL SETUSER(CUSER);
 146   3            END SETCUSER;

 147   2      setduser: procedure;
 148   3            call setuser(odest.user);
 149   3            end setduser;

 150   2      SETSUSER: PROCEDURE;
 151   3            CALL SETUSER(source.user);
 152   3            END SETSUSER;

 153   2      RD$RANDOM: PROCEDURE(FCB) BYTE;
 154   3            DECLARE FCB ADDRESS;
                /** $if mpm **/
 155   3            call retcodes(mon3(33,fcb));
                /** $else **/
                /** $endif **/
 156   3            return dcnt;
 157   3            END RD$RANDOM;

 158   2      write$random: procedure(fcb) byte;
 159   3            declare fcb address;
                /** $if mpm **/
 160   3            call retcodes(mon3(34,fcb));
                /** $else **/
                /** $endif **/
 161   3            return dcnt;
 162   3            end write$random;

 163   2      retfsize: procedure(fcb) byte;
 164   3            declare fcb address;
 165   3            return mon2(35,fcb);
 166   3            end retfsize;

 167   2      SET$RANDOM: PROCEDURE(FCB);
 168   3            DECLARE FCB ADDRESS;
                     /* SET RANDOM RECORD POSITION */
 169   3            CALL MON1(36,FCB);
 170   3            END SET$RANDOM;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
            /** $if mpm **/
171   2     multsect: procedure(cnt);
172   3         declare cnt byte;
173   3         if mseccnt <> cnt then
174   3             call mon1(44,(mseccnt := cnt));
175   3         end multsect;

176   2     flushbuf: procedure;
177   3         call mon1(48, 0ffh);    /* 0FFH = flush and discard buffers */
178   3         end flushbuf;

179   2     conatlst: procedure byte;
180   3         return mon2(161,0);
181   3         end conatlst;
            /** $endif **/


182   2     MOVE: PROCEDURE(S,D,N);
183   3         DECLARE (S,D) ADDRESS, N BYTE;
184   3         DECLARE A BASED S BYTE, B BASED D BYTE;
185   3             DO WHILE (N:=N-1) <> 255;
186   4             B = A; S = S+1; D = D+1;
189   4             END;
190   3         END MOVE;

            /* errtype error messages */
191   2         declare er00(*) byte data ('DISK READ$');
192   2         declare er01(*) byte data ('DISK WRITE$');
193   2         declare er02(*) byte data ('VERIFY$');
194   2         declare er03(*) byte data ('INVALID DESTINATION$');
195   2         declare er04(*) byte data ('INVALID SOURCE$');
196   2         declare er05(*) byte data ('USER ABORTED$');
197   2         declare er06(*) byte data ('BAD PARAMETER$');
198   2         declare er07(*) byte data ('INVALID USER NUMBER$');
199   2         declare er08(*) byte data ('INVALID FORMAT$');
200   2         declare er09(*) byte data ('HEX RECORD CHECKSUM$');
201   2         declare er10(*) byte data ('FILE NOT FOUND$');
202   2         declare er11(*) byte data ('START NOT FOUND$');
203   2         declare er12(*) byte data ('QUIT NOT FOUND$');
204   2         declare er13(*) byte data ('INVALID HEX DIGIT$');
205   2         declare er14(*) byte data ('CLOSE FILE$');
206   2         declare er15(*) byte data ('UNEXPECTED END OF HEX FILE$');
207   2         declare er16(*) byte data ('INVALID SEPARATOR$');
208   2         declare er17(*) byte data ('NO DIRECTORY SPACE$');
209   2         declare er18(*) byte data ('INVALID FORMAT WITH SPARCE FILE$');
            /** $if mpm **/
210   2         declare er19(*) byte data ('MAKE FILE$');
211   2         declare er20(*) byte data ('OPEN FILE$');
212   2         declare er21(*) byte data ('PRINTER BUSY$');
213   2         declare er22(*) byte data ('CAN''T DELETE TEMP FILE$');
            /** $endif **/

214   2         declare errmsg(*) address data(
                    .er00,.er01,.er02,.er03,.er04,
                    .er05,.er06,.er07,.er08,.er09,
                    .er10,.er11,.er12,.er13,.er14,
                    .er15,.er16,.er17,.er18
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
            /** $if mpm **/
                  ,.er19,.er20,.er21,.er22
            /** $endif **/
                  );

215   2       declare sper00(*) byte data ('NO DIRECTORY SPACE$');
216   2       declare sper01(*) byte data ('NO DATA BLOCK$');
217   2       declare sper02(*) byte data ('CAN''T CLOSE CURRENT EXTENT$');
218   2       declare sper03(*) byte data ('SEEK TO UNWRITTEN EXTENT$');
219   2       declare sper05(*) byte data ('RANDOM RECORD OUT OF RANGE$');
220   2       declare sper06(*) byte data ('RECORDS DON''T MATCH$');
221   2       declare sper07(*) byte data ('RECORD LOCKED$');
222   2       declare sper08(*) byte data ('INVALID FILENAME$');
223   2       declare sper09(*) byte data ('FCB CHECKSUM$');

224   2       declare numspmsgs lit '10';  /* number of extended messages */
225   2       declare special$msg(numspmsgs) address data(
                    .sper00,.sper01,.sper02,.sper03,.sper00,
                    .sper05,.sper06,.sper07,.sper08,.sper09);

            /** $if mpm **/
                /* extended error messages */
226   2       declare ex00(*) byte data ('$');  /* NO MESSAGE */
227   2       declare ex01(*) byte data ('NONRECOVERABLE$');
228   2       declare ex02(*) byte data ('R/O DISK$');
229   2       declare ex03(*) byte data ('R/O FILE$');
230   2       declare ex04(*) byte data ('INVALID DISK SELECT$');
231   2       declare ex05(*) byte data ('INCOMPATIBLE MODE$');
232   2       declare ex07(*) byte data ('INVALID PASSWORD$');
233   2       declare ex08(*) byte data ('ALREADY EXISTS$');
234   2       declare ex10(*) byte data ('LIMIT EXCEEDED$');

235   2       declare nummsgs lit '11';  /* number of extended messages */
236   2       declare extmsg(nummsgs) address data(
                    .ex00,.ex01,.ex02,.ex03,.ex04,
                    .ex05,.sper09,.ex07,.ex08,.sper08,
                    .ex10);
            /** $endif **/

237   2     error$cleanup: procedure;
            /** $if mpm **/
238   3         call multsect(1);
            /** $endif **/
239   3         eretry = 0;   /* initialize to no error retry */
240   3         if opened then   /* if source file opened */
241   3             do; call setsuser;
243   4             call close(.source);
244   4             opened = false;
245   4             end;
246   3         if made then
247   3             do; call setduser;
249   4             call close(.dest);
250   4             call delete(.dest);  /* delete destination scratch file */
251   4             end;
                  /* Zero the command length in case this is a single command */
252   3             comlen = 0;
253   3             retry = true;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
254  3          call print(.('ERROR: $'));
255  3       end error$cleanup;

256  2     error: procedure (errtype);
257  3       declare errtype byte;

258  3       call error$cleanup;
259  3       call printx(errmsg(errtype));
260  3       call crlf;
261  3       go to reset;
262  3       end error;

263  2     xerror: procedure (funcno,fileadr);
264  3       declare temp    byte,
                  i       byte,
                  sdcnt   byte,
                  sexten  byte,
                  funcno  byte,
                  fileadr address,
                  fcb based fileadr (fsize) byte;

265  3       declare message$index$tbl(17) byte data
                  (2,18,13,15,9,3,10,20,14,10,22,17,19,0,1,0,1);

266  3       sdcnt = dcnt;
267  3       sexten = exten;
268  3       call error$cleanup;

269  3       if (funcno < 6) or (sdcnt <> 0ffh) then
270  3          sexten = 0;
271  3       else sexten = sexten and 0fh;

272  3       call printx(errmsg(message$index$tbl(funcno)));

273  3       if (funcno > 12) and (funcno < 17) and
             (sdcnt <> 0ffh) and (sdcnt <= numspmsgs) then
274  3          do; call printchar(' ');
276  4          call printx(special$msg(sdcnt-1));
277  4          sexten = 0;
278  4          end;

           /** $if mpm **/
279  3       if sexten < nummsgs then
280  3          do; call printchar(' ');
282  4          call printx(extmsg(sexten));
283  4          end;
           /** $endif **/

284  3       call printx(.(' - $'));
285  3       if fileadr <> 0 then
286  3           do; call printchar('A' + fcb(0) - 1);
288  4           call printchar(':');
289  4               do i = 1 to fnsize;
290  5               if (temp := fcb(i) and 07fh) <> ' ' then
291  5                   do; if i = fext then call printchar('.');
294  6                   call printchar(temp);
295  6                   end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
296  5              end;
297  4            end;
298  3         call crlf;

299  3         if (sdcnt = 3) or (sdcnt = 4) or (sdcnt = 6) or (sdcnt = 8) then
300  3            eretry = ambig;
                  else
301  3            if (sexten = 3) or ((sexten > 4) and (sexten < 9)) or (sexten > 9) then
302  3                 eretry = ambig;

                  go to reset;
304  3         end xerror;

305  2      FORMERR: PROCEDURE;
306  3         call error(8);  /* invalid format */
307  3         END FORMERR;

308  2      CONBRK: PROCEDURE;
               /* CHECK CONSOLE CHARACTER READY */
309  3         if mon2(11,0) <> 0 then
310  3             if mon2(6,0fdh) = cntrlc then
311  3                call error(5);
312  3         END CONBRK;

313  2      MAXSIZE: procedure byte;
               /* three byte compare of random record field
                  returns true if source.fcb.ranrec >= filesize */

314  3         if (source.fcb(35) < filsize(2)) then
315  3           return false;
316  3         if (source.fcb(35) = filsize(2)) then
317  3           do;
318  4           if (source.fcb(34) < filsize(1)) then
319  4             return false;
320  4           if (source.fcb(34) = filsize(1)) then
321  4             do;
322  5             if (source.fcb(33) < filsize(0)) then
323  5               return false;
324  5             end;
325  4           end;
326  3         return true;
327  3         end maxsize;

328  2      SETUPDEST: PROCEDURE;
329  3         call setduser;  /* destination user */
              /** $if mpm **/
330  3         call move(.odest,.dest,(frsize + nsize + 1));  /* save original dest */
              /** $else **/
              /** $endif **/
               /* MOVE THREE CHARACTER EXTENT INTO DEST FCB */
331  3         CALL MOVE(.('$$$'),.DEST.FCB(FEXT),FEXTL);
              /** $if mpm **/
332  3         odest.fcb(6) = odest.fcb(6) or 80h;
333  3         call open(.odest);  /* try to open destination file */
334  3         odcnt = dcnt;       /* and save error code */
335  3         if odcnt <> 255 then
336  3             call close(.odest);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
337  3          else if (exten and 0fh) <> 0 then /* file exists */
338  3              call xerror(7,.odest);       /* but can't open - error */

                CALL DELETE(.DEST); /* REMOVE OLD $$$ FILE */
340  3          if dcnt = 255 and exten <> 0 then
                    /* cant delete temp file */
341  3              call xerror(10,.dest);
342  3          CALL MAKE(.DEST); /* CREATE A NEW ONE */
343  3          IF DCNT = 255 THEN
344  3              if (exten and 0fh) = 0 then
345  3                  call xerror(11,.dest);  /* no directory space */
346  3              else call xerror(12,.dest);  /* make file error */
           /** $else **/
           /** $endif **/
347  3          DEST.FCB(32) = 0;
348  3          made = true;
349  3          END SETUPDEST;

350  2      SETUPSOURCE: PROCEDURE;
351  3          declare (i,j) byte;
352  3          CALL SETSUSER; /* SOURCE USER */
           /** $if mpm **/
353  3          source.fcb(6) = source.fcb(6) or 80h;
           /** $endif **/
354  3          CALL OPEN(.SOURCE);  /* open source */
355  3          if dcnt <> 255 then
356  3              opened = true;
357  3          IF (NOT RSYS) AND ROL(SOURCE.FCB(10),1) THEN
                    /* skip system file */
358  3              DCNT = 255;
359  3          IF DCNT = 255 THEN
           /** $if mpm **/
360  3              if (exten and 0fh) = 0 then
361  3                  call xerror(6,.source);  /* file not found */
                    else
362  3                  call xerror(7,.source);  /* open file error */
           /** $else **/
           /** $endif **/
363  3          f1 = source.fcb(1) and 80h;  /* save file atributes */
364  3          f2 = source.fcb(2) and 80h;
365  3          f3 = source.fcb(3) and 80h;
366  3          f4 = source.fcb(4) and 80h;
367  3          ro = source.fcb(9) and 80h;
368  3          sys = source.fcb(10) and 80h;
369  3          dcnt = retfsize(.source);
370  3          call move(.source.fcb(33),.filsize,3);
371  3          SOURCE.FCB(32) = 0;
372  3          source.fcb(33),source.fcb(34),source.fcb(35) = 0;
                /* cause immediate read with no preceding write */
373  3          NSOURCE = 0ffffh;
374  3          END SETUPSOURCE;

375  2      WRITEDEST: PROCEDURE;
                /* WRITE OUTPUT BUFFERS UP TO BUT NOT INCLUDING POSITION
                NDEST - THE LOW ORDER 7 BITS OF NDEST ARE ZERO */
376  3          DECLARE (J,DATAOK) BYTE,
                    (tdest,n)      address;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
377  3          if not made then call setupdest;
379  3          if (n != ndest and 0ff80h) = 0 then return;
381  3          tdest = 0;
382  3          call setduser;  /* destination user */
383  3          if (sparfil != (sparfil or insparc)) then
                    /* set up fcb from random record no. */
384  3              do;
                /** $if mpm **/
385  4              call multsect(1);
                /** $endif **/
386  4              CALL SETDMA(.dbuff(tdest));
387  4              if write$random(.dest) <> 0 then
388  4                  call xerror(16,.dest);  /* DISK WRITE ERROR */
389  4              end;
                else
390  3              CALL SETRANDOM(.DEST); /* SET BASE RECORD FOR VERIFY */
                /** $if mpm **/
391  3          if fastcopy then
392  3              do; bufsize = maxmbuf;
394  4              call multsect(maxmcnt);
395  4              end;
                else
396  3              do; bufsize = 128;
398  4              call multsect(1);
399  4              end;
                /** $endif **/

400  3              do while n - tdest > 127;
                /** $if mpm **/
401  4              if fastcopy and (n - tdest < maxmbuf) then
402  4                  do; bufsize = n - tdest;
404  5                  call multsect(low(shr(bufsize,7)));
405  5                  end;
                /** $endif **/
                    /* SET DMA ADDRESS TO NEXT BUFFER */
406  4              CALL SETDMA(.dbuff(tdest));
407  4              call diskwrite(.dest);
408  4              IF dcnt <> 0 THEN
409  4                  call xerror(14,.dest);  /* DISK WRITE ERROR */
                /** $if mpm **/
410  4              tdest = tdest + bufsize;
                /** $else **/
                /** $endif **/
411  4              END;

412  3          IF VERIF THEN /* VERIFY DATA WRITTEN OK */
413  3              DO;
414  4              call flushbuf;
415  4              tdest = 0;
                /** $if mpm **/
416  4              call multsect(1);
                /** $endif **/
417  4              CALL SETDMA(.BUFF); /* FOR COMPARE */
418  4                  do while tdest < n;
419  5                  DATAOK = (RDRANDOM(.DEST) = 0);
420  5                  if (DESTR != DESTR + 1) = 0 then  /* 3 byte inc for */
421  5                      destr2 = destr2 + 1;  /* next random record */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
422  5                J = 0;
                          /* PERFORM COMPARISON */
423  5                    DO WHILE DATAOK AND J < 80H;
424  6                    DATAOK = (BUFF(J) = DBUFF(tdest+J));
425  6                    J = J + 1;
426  6                    END;
427  5                tdest = tdest + 128;
428  5                IF NOT DATAOK THEN
429  5                    call xerror(0,.dest);  /* VERIFY ERROR */
430  5                END;
431  4            call diskrd(.dest);
                  /* NOW READY TO CONTINUE THE WRITE OPERATION */
432  4            END;
433  3        CALL SETRANDOM(.DEST); /* set base record for sparce copy */
434  3        call move(.dbuff(tdest),.dbuff(0),low(ndest := ndest - tdest));
435  3        END WRITEDEST;

436  2    FILLSOURCE: PROCEDURE;
          /* FILL THE SOURCE BUFFER */
437  3        call conbrk;
         /** $if mpm **/
438  3        if fastcopy then
439  3            do; bufsize = maxmbuf;
441  4            call multsect(maxmcnt);
442  4            end;
443  3        else do;
444  4            bufsize = 128;
445  4            call multsect(1);
446  4            end;
         /** $endif **/
447  3        CALL SETSUSER; /* SOURCE USER NUMBER SET */
448  3        nsource = nsbuf;
449  3            do while sblen - nsbuf > 127;
450  4            if fastcopy and (sblen - nsbuf < maxmbuf) then
451  4                do; bufsize = (sblen - nsbuf) and 0ff80h;
453  5                call multsect(low(shr(bufsize,7)));
454  5                end;
                  /* SET DMA ADDRESS TO NEXT BUFFER POSIITION */
455  4            CALL SETDMA(.SBUFF(nsbuf));
456  4            extsave = source.fcb(12);  /* save extent field */
457  4            call diskrd(.source);
458  4            IF dcnt <> 0 THEN
459  4                DO; IF dcnt <> 1 THEN
461  5                    call xerror(13,.source);  /* DISK READ ERROR */
                      /* END - OF - FILE */
         /** $if mpm **/
462  5                if fastcopy then   /* add no. sectors copied */
463  5                    nsbuf = nsbuf + shl(double(exten),7);
                      /* nsbuf = nsbuf + shl(double(exten and 0f0h),3); */
         /** $endif **/
                      /* check boundry condition for bug in bdos and correct */
464  5                if (source.fcb(12) <> extsave) and (source.fcb(32) = 80h) then
465  5                    source.fcb(32) = 0;  /* zero current record */
466  5                call set$random(.source);
467  5                if (insparc != not maxsize) then
468  5                    do;
469  6                    if concat or (not fastcopy) then
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                            /* invalid format with sparce file */
470   6                     call xerror(1,.source);
471   6                  end;
                        else
472   5                  do;
473   6                  call close(.source);
474   6                  opened = false;
475   6                  end;
476   5               endofsrc = true;  /* set end of source file */
477   5               SBUFF(nsbuf) = ENDFILE; re turn;
479   5               END;
                    ELSE
                /** $if mpm **/
480   4               nsbuf = nsbuf + bufsize;
                /** $else **/
                /** $endif **/
481   4           END;
482   3       END FILLSOURCE;

483   2       PUTDCHAR: PROCEDURE(B);
484   3           DECLARE B BYTE;
                    /* WRITE BYTE B TO THE DESTINATION DEVICE GIVEN BY ODEST.TYPE */
485   3           IF B >= ' ' THEN
486   3               DO; COLUMN = COLUMN + 1;
488   4               IF DELET > 0 THEN /* MAY BE PAST RIGHT SIDE */
489   4                   DO; IF COLUMN > DELET THEN RETURN;
492   5                   END;
493   4               END;
494   3           if echo then call mon1(2,b);  /* echo to console */
496   3           do case odest.type;
                        /* CASE 0 IS OUT */
497   4                   CALL OUTD(B);
                        /* CASE 1 IS PRN, TABS EXPANDED, LINES LISTED */
498   4                   call mon1(5,b);
                        /* CASE 2 IS LST */
499   4                   CALL MON1(5,B);
                        /* CASE 3 IS axo */
500   4       axocase:
                /** $if not mpm **/
                        CALL MON1(4,B);
                /** $else **/
                /** $endif **/
                        /* CASE 4 IS DESTINATION FILE */
501   4                   DO;
502   5                   IF NDEST >= DBLEN THEN CALL WRITEDEST;
504   5                   DBUFF(NDEST) = B;
505   5                   NDEST=NDEST+1;
506   5                   END;
                        /* CASE 5 IS AUX */
507   4                   goto axocase;
                        /* CASE 6 IS CON */
508   4                   CALL MON1(2,B);
509   4               END; /* of case */
510   3           END PUTDCHAR;

511   2       PUTDESTC: PROCEDURE(B);
512   3           DECLARE (B,I) BYTE;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                /* WRITE DESTINATION CHARACTER, TAB EXPANSION */
513   3         IF B <> TAB THEN CALL PUTDCHAR(B);
515   3         ELSE IF TABS = 0 THEN CALL PUTDCHAR(B);
                ELSE  /* B IS TAB CHAR, TABS > 0 */
517   3             DO; I = COLUMN;
519   4                 DO WHILE I >= TABS;
520   5                 I = I - TABS;
521   5                 END;
522   4             I = TABS - I;
523   4                 DO WHILE I > 0;
524   5                 I = I - 1;
525   5                 CALL PUTDCHAR(' ');
526   5                 END;
527   4             END;
528   3         IF B = CR THEN COLUMN = 0;
530   3         END PUTDESTC;

531   2     PRINT1: PROCEDURE(B);
532   3         DECLARE B BYTE;
533   3         IF (ZEROSUP := ZEROSUP AND B = 0) THEN
534   3             CALL PUTDESTC(' ');
                ELSE
535   3             CALL PUTDESTC('0'+B);
536   3         END PRINT1;

537   2     PRINTDIG: PROCEDURE(D);
538   3         DECLARE D BYTE;
539   3         CALL PRINT1(SHR(D,4)); CALL PRINT1(D AND 1111B);
541   3         END PRINTDIG;

542   2     NEWLINE: PROCEDURE;
543   3         DECLARE ONE BYTE;
544   3         ONE = 1;
545   3         ZEROSUP = (NUMB = 1);
546   3         C1 = DEC(C1+ONE); C2 = DEC(C2 PLUS 0); C3 = DEC(C3 PLUS 0);
549   3         CALL PRINTDIG(C3); CALL PRINTDIG(C2); CALL PRINTDIG(C1);
552   3         IF NUMB = 1 THEN /* USUALLY PRINTER OUTPUT */
553   3             DO; CALL PUTDESTC(':'); CALL PUTDESTC(' ');
556   4             END;
                ELSE
557   3             CALL PUTDESTC(TAB);
558   3         END NEWLINE;

559   2     PUTDEST: PROCEDURE(B);
560   3         DECLARE (I,B) BYTE;
                /* WRITE DESTINATION CHARACTER, CHECK TABS AND LINES */
561   3         IF FORMF THEN /* SKIP FORM FEEDS */
562   3             DO; IF B = FF THEN RETURN;
565   4             END;
566   3         IF PUTNUM THEN /* END OF LINE OR START OF FILE */
567   3             DO;
568   4             IF (B <> FF) and (b <> endfile) THEN
569   4                 DO;    /* NOT FORM FEED or end of file */
570   5                 IF (I:=PAGCNT) <> 0 THEN /* PAGE EJECT */
571   5                     DO; IF I=1 THEN I=LPP;
574   6                     IF (LINENO := LINENO + 1) >= I THEN
575   6                         DO; LINENO = 0; /* NEW PAGE */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
577  7                      CALL PUTDESTC(FF);
578  7                      END;
579  6                  END;
580  5              IF NUMB > 0 THEN
581  5                  CALL NEWLINE;
582  5              PUTNUM = FALSE;
583  5              END;
584  4          END;
585  3      IF B = FF THEN LINENO = 0;
587  3      CALL PUTDESTC(B);
588  3      IF B = LF THEN PUTNUM = TRUE;
590  3      END PUTDEST;


591  2  UTRAN: PROCEDURE(B) BYTE;
592  3      DECLARE B BYTE;
           /* TRANSLATE ALPHA TO UPPER CASE */
593  3      IF B >= 110$0001B AND B <= 111$1010B THEN /* LOWER CASE */
594  3          B = B AND 101$1111B; /* TO UPPER CASE */
595  3      RETURN B;
596  3      END UTRAN;

597  2  LTRAN: PROCEDURE(B) BYTE;
598  3      DECLARE B BYTE;
           /* TRANSLATE TO LOWER CASE ALPHA */
599  3      IF B >= 'A' AND B <= 'Z' THEN
600  3          B = B OR 10$0000B; /* TO LOWER */
601  3      RETURN B;
602  3      END LTRAN;

603  2  GETSOURCEC: PROCEDURE BYTE;
           /* READ NEXT SOURCE CHARACTER */
604  3      DECLARE (B,CONCHK) BYTE;

605  3      CONCHK = TRUE; /* CONSOLE STATUS CHECK BELOW */
606  3          DO CASE source.type;
               /* CASE 0 IS out */
607  4              go to notsource;
               /* CASE 1 IS prn */
608  4              go to notsource;
               /* CASE 2 IS 1st */
609  4              notsource:
                       call error(4);  /* INVALID SOURCE */
               /* CASE 3 IS axo */
610  4              go to notsource;
               /* CASE 4 IS SOURCE FILE */
611  4              DO;
612  5              IF NSOURCE >= SBLEN THEN
613  5                  do; if dblbuf or (not dfile) then
615  6                      nsbuf = 0;
616  6                  else if (nsource <> 0ffffh) then
617  6                      do; call writedest;
619  7                      nsbuf = ndest;
620  7                      end;
                       CALL FILLSOURCE;
622  6                  end;
623  5              B = SBUFF(NSOURCE);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
624  5                NSOURCE = NSOURCE + 1;
625  5                END;
                    /* CASE 5 IS AUX */
626  4                goto axicase;
                    /* CASE 6 IS CON */
627  4                DO; CONCHK = FALSE; /* DON'T CHECK CONSOLE STATUS */
629  5                B = MON2(1,0);
630  5                END;
                    /* CASE 7 IS axi */
631  4           axicase:
                 /** $if not mpm **/
                    B = MON2(3,0) AND 7FH;
                 /** $else **/
                 /** $endif **/
                    /* CASE 7 IS INP */
632  4                B = INPD;
633  4               END; /* OF CASES */

634  3            IF CONCHK THEN /* TEST FOR CONSOLE CHAR READY */
635  3               DO;
636  4               IF obj THEN /* SOURCE IS AN OBJECT FILE */
637  4                  CONCHK = ((CONCNT := CONCNT + 1) = 0);
                    ELSE /* ASCII */
638  4                  CONCHK = (B = LF);
639  4               IF CONCHK THEN
640  4                  DO;
641  5                  call CONBRK;
642  5                  END;
643  4               END;
644  3            IF ZEROP THEN B = B AND 7FH;
646  3            IF UPPER THEN RETURN UTRAN(B);
648  3            IF LOWER THEN RETURN LTRAN(B);
650  3            RETURN B;
651  3            END GETSOURCEC;

652  2           GETSOURCE: PROCEDURE BYTE;
                    /* GET NEXT SOURCE CHARACTER */
653  3               DECLARE CHAR BYTE;
654  3               MATCH: PROCEDURE(B) BYTE;
                        /* MATCH START AND QUIT STRINGS */
655  4                   DECLARE (B,C) BYTE;
656  4                   IF (C:=COMBUFF(B:=(B+MATCHLEN))) = ENDFILE THEN /* END MATCH */
657  4                      DO; COMBUFF(B) = CHAR; /* SAVE CURRENT CHARACTER */
659  5                      RETURN TRUE;
660  5                      END;
661  4                   IF C = CHAR THEN MATCHLEN = MATCHLEN + 1;
                        ELSE
663  4                      MATCHLEN = 0; /* NO MATCH */
664  4                   RETURN FALSE;
665  4                   END MATCH;

666  3               IF QUITLEN > 0 THEN
667  3                   DO; IF (QUITLEN := QUITLEN - 1) = 1 THEN RETURN LF;
670  4                   RETURN ENDFILE; /* TERMINATED WITH CR,LF,ENDFILE */
671  4                   END;
672  3               DO FOREVER; /* LOOKING FOR START */
673  4               IF FEEDLEN > 0 THEN /* GET SEARCH CHARACTERS */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
674  4            DO; FEEDLEN = FEEDLEN - 1;
676  5            CHAR = COMBUFF(FEEDBASE);
677  5            FEEDBASE = FEEDBASE + 1;
678  5            RETURN CHAR;
679  5            END;
680  4        IF (CHAR := GETSOURCEC) = ENDFILE THEN RETURN ENDFILE;
682  4        IF STARTS > 0 THEN /* LOOKING FOR START STRING */
683  4            DO; IF MATCH(STARTS) THEN
685  5                DO; FEEDBASE = STARTS; STARTS = 0;
688  6                FEEDLEN = MATCHLEN + 1;
689  6                matchlen = 0;
690  6                END; /* OTHERWISE NO MATCH, SKIP CHARACTER */
691  5            END;
692  4        ELSE IF QUITS > 0 THEN /* PASS CHARACTERS TIL MATCH */
693  4            DO; IF MATCH(QUITS) THEN
695  5                DO; QUITS = 0; QUITLEN = 2;
                         /* SUBSEQUENTLY RETURN CR, LF, ENDFILE */
698  6                RETURN CR;
699  6                END;
700  5            RETURN CHAR;
701  5            END;
                 ELSE
702  4            RETURN CHAR;
703  4        END; /* OF DO FOREVER */
704  3        END GETSOURCE;

705  2    RD$EOF: PROCEDURE BYTE;
             /* RETURN TRUE IF END OF FILE */
706  3        CHAR = GETSOURCE;
707  3        IF obj THEN RETURN (endofsrc and (nsource > nsbuf));
709  3        RETURN (CHAR = ENDFILE);
710  3        END RD$EOF;


711  2    HEXRECORD: PROCEDURE;
712  3        DECLARE (h, hbuf, RL, CS, RT) BYTE,
                     zerorec byte,  /* true if last record had length of zero */
                     LDA ADDRESS;  /* LOAD ADDRESS WHICH FOLLOWS : */

713  3        ckhex: procedure byte;
714  4            IF H - '0' <= 9 THEN
715  4                RETURN H-'0';
716  4            IF H - 'A' > 5 THEN
717  4                CALL xerror(2,,source);  /* invalid hex digit */
718  4            RETURN H - 'A' + 10;
719  4            end ckhex;

720  3        rdhex: procedure byte;
721  4            call putdest(h := getsource);
722  4            return ckhex;
723  4            end rdhex;

724  3        RDCS: PROCEDURE BYTE;
                     /* READ BYTE WITH CHECKSUM */
725  4            RETURN CS := CS + (SHL(RDHEX,4) OR RDHEX);
726  4            END RDCS;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
727  3          RDADDR: PROCEDURE ADDRESS;
                   /* READ DOUBLE BYTE WITH CHECKSUM */
728  4             RETURN SHL(DOUBLE(RDCS),8) OR RDCS;
729  4             END RDADDR;

                /* READ HEX FILE AND CHECK EACH RECORD
                FOR VALID DIGITS, AND PROPER CHECKSUM */
730  3          zerorec = false;
                /* READ NEXT RECORD */
731  3          h = getsource;
732  3             do forever;
                   /* SCAN FOR THE ':' */
733  4                DO WHILE h <> ':';
734  5                IF (h = ENDFILE) THEN
735  5                   do; if zerorec then return;
738  6                   CALL xerror(3,,source);  /* unexpected end of hex file */
739  6                   end;
740  5                call putdest(h);
741  5                h = getsource;
742  5                END;

                   /* ':' FOUND */
                   /* check for end of hex record */
743  4             h = getsource;
744  4             rl = shl(ckhex,4);
745  4             hbuf = h; h = getsource;
747  4             rl = rl or ckhex;
748  4             if (rl = 0) then zerorec = true;
750  4             else zerorec = false;
751  4             if (zerorec and ignor) then
752  4                do while (h <> ':') and (h <> endfile);
753  5                h = getsource;
754  5                end;
755  4             else do; call putdest(':');
757  5                call putdest(hbuf);
758  5                call putdest(h);
759  5                cs = rl;
760  5                LDA = RDADDR; /* LOAD ADDRESS */

                   /* READ WORDS UNTIL RECORD LENGTH EXHAUSTED */
761  5                RT = RDCS; /* RECORD TYPE */
762  5                   DO WHILE RL <> 0; RL = RL - 1;
764  6                   hbuf = RDCS;
                      /* INCREMENT LA HERE FOR EXACT ADDRESS */
765  6                   END;

                   /* CHECK SUM */
766  5                IF rdcs <> 0 THEN
767  5                   CALL xerror(4,,source);  /* hex record checksum */
768  5                h = getsource;
769  5                end;
770  4             end; /* do forever */
771  3          END HEXRECORD;

772  2       CK$STRINGS: PROCEDURE;
773  3          IF STARTS > 0 THEN
774  3             call error(11);  /* START NOT FOUND */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
775  3          IF QUITS  > 0 THEN
776  3             call error(12);  /* QUIT NOT FOUND */
777  3          END CK$STRINGS;

778  2      CLOSEDEST: PROCEDURE;
779  3             DO WHILE (LOW(NDEST) AND 7FH) <> 0;
780  4             CALL PUTDEST(ENDFILE);
781  4             END;
782  3          CALL CK$STRINGS;
783  3          CALL WRITEDEST;
784  3          call setduser;  /* destination user */
785  3          CALL CLOSE(.DEST);
786  3          IF DCNT = 255 THEN
           /** $if mpm **/
787  3             call xerror(8,.dest);  /* CLOSE FILE */
788  3          IF odcnt <> 255 THEN /* FILE EXISTS */
789  3             do;
           /** $else **/
           /** $endif **/
790  4             IF ROL(odest.fcb(9),1) THEN /* READ ONLY */
791  4                DO;
792  5                IF NOT WRROF THEN
793  5                   DO;
794  6                      do while ((dcnt <> 'Y') and (dcnt <> 'N'));
795  7                      CALL PRINT (.('DESTINATION IS R/O, DELETE (Y/N)? $'));
796  7                      dcnt = utran(rdchar);
797  7                      end;
798  6                   IF dcnt <> 'Y' THEN
799  6                      DO; CALL PRINT(.('**NOT DELETED**$'));
801  7                      CALL CRLF;
802  7                      CALL DELETE(.DEST);
803  7                      RETURN;
804  7                      END;
805  6                   CALL CRLF;
806  6                   END;
807  5                END;
                   /* reset r/o and sys attributes */
808  4                odest.fcb(9) = odest.fcb(9) and 7fh;
809  4                odest.fcb(10) = odest.fcb(10) AND 7FH;
810  4                CALL SETIND(.odest);
811  4                CALL DELETE(.odest);
812  4                END;
813  3          CALL MOVE(.odest.fcb,.dest.fcb(16),16); /* READY FOR RENAME */
814  3          CALL RENAME(.DEST);
           /* set destination attributes same as source */
815  3          odest.fcb(1) = (odest.fcb(1) and 07fh) or f1;
816  3          odest.fcb(2) = (odest.fcb(2) and 07fh) or f2;
817  3          odest.fcb(3) = (odest.fcb(3) and 07fh) or f3;
818  3          odest.fcb(4) = (odest.fcb(4) and 07fh) or f4;
819  3          odest.fcb(8) = (odest.fcb(8) and 07fh);
820  3          odest.fcb(9) = (odest.fcb(9) and 07fh) or ro;
821  3          odest.fcb(10) = (odest.fcb(10) and 07fh) or sys;
822  3          odest.fcb(11) = (odest.fcb(11) and 07fh);
823  3          call setind(.odest);
824  3          if archiv then   /* set archive bit */
825  3             do; call setsuser;
827  4             source.fcb(11) = source.fcb(11) or 080h;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
828  4            source.fcb(12) = 0;
829  4            call setind(.source);
830  4            end;
831  3        END CLOSEDEST;

832  2    SIZE$MEMORY: PROCEDURE;
              /* SET UP SOURCE AND DESTINATION BUFFERS */
833  3        if not dblbuf then
834  3            do;  /* ABSORB THE SOURCE BUFFER INTO THE DEST BUFFER */
835  4            sbase = .memory;
836  4            sblen,dblen = ((maxb - .memory) and 0ff80h) - 128;
837  4            end;
838  3        else do;  /* may need to write destination buffer */
839  4            sblen,dblen = (shr((maxb - .memory),1) and 0ff80h) - 128;
840  4            sbase = .memory + dblen + 128;
841  4            if ndest >= dblen then call writedest;
843  4            nsbuf = 0;
844  4            end;
845  3        END SIZE$MEMORY;

846  2    setupeob: procedure;
              /* sets nsbuf to end of source buffer */
847  3        declare i byte;
848  3        if (not obj) and (nsbuf <> 0) then
849  3            do; tblen = nsbuf - 128;
851  4                do i = 0 to 128;
852  5                if (sbuff(tblen + i)) = endfile then
853  5                    do; nsbuf = tblen + i;
855  6                    return;
856  6                    end;
857  5                end;
858  4            end;
859  3        end setupeob;

860  2    SIMPLECOPY: PROCEDURE;
861  3        DECLARE I BYTE;
862  3        declare
                 fast lit '0',  /* fast file to file copy */
                 chrt lit '1',  /* character transfer option */
                 dubl lit '2';  /* double buffer required for file copy */
863  3        declare optype(26) byte data (
              /* option type for each option character */
                 fast,  /* for A option */
                 fast,  /* for B option */
                 fast,  /* for C option */
                 dubl,  /* for D option */
                 chrt,  /* for E option */
                 dubl,  /* for F option */
                 fast,  /* for G option */
                 chrt,  /* for H option */
                 dubl,  /* for I option */
                 fast,  /* for J option */
                 fast,  /* for K option */
                 chrt,  /* for L option */
                 fast,  /* for M option */
                 dubl,  /* for N option */
                 fast,  /* for O option */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                    dubl,  /* for P option */
                    dubl,  /* for Q option */
                    fast,  /* for R option */
                    dubl,  /* for S option */
                    dubl,  /* for T option */
                    chrt,  /* for U option */
                    fast,  /* for V option */
                    fast,  /* for W option */
                    fast,  /* for X option */
                    fast,  /* for Y option */
                    chrt); /* for Z option */

864   3         chkrandom: procedure;
865   4             call setsuser;
866   4             call set$random(.source);
                /** $if mpm **/
867   4             call multsect(1);
                /** $endif **/
868   4             call setdma(.buff);
869   4                 do forever;
870   5                 if (((dcnt := rd$random(.source)) = 0) or maxsize) then
871   5                     do; destr = sourcer;
873   6                     destr2 = sourcer2;
874   6                     endofsrc = false;
875   6                     return;
876   6                     end;
877   5                 if dcnt = 1 then
878   5                     do; if (sourcer := sourcer + 1) = 0 then
880   6                         sourcer2 = sourcer2 + 1;
881   6                     end;
882   5                 else if dcnt = 4 then
883   5                     do;
884   6                     if (sourcer := (sourcer + 128) and 0ff80h) = 0 then
885   6                         sourcer2 = sourcer2 + 1;
886   6                     end;
                      else
887   5                     call xerror(15,,source);
888   5                 end;
889   4             end chkrandom;

890   3         fastcopy = (sfile and dfile);
891   3         endofsrc = false;
892   3         dblbuf = false;
893   3         sparfil = false;
894   3         insparc = false;
                    /* LOOK FOR PARAMETERS */
895   3             DO I = 0 TO 25;
896   4             IF CONT(I) <> 0 THEN
897   4                 DO;
898   5                 IF optype(i) = chrt THEN
899   5                     FASTCOPY = FALSE;
                      else
900   5                 if optype(i) = dubl then
901   5                     do; dblbuf = (sfile and dfile);
903   6                     fastcopy = false;
904   6                     end;
                      END;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
906   4           END;

907   3       CALL SIZE$MEMORY;
908   3       if sfile then
909   3           CALL SETUPSOURCE;
                  /* FILES READY FOR COPY */

910   3       if fastcopy then
911   3           do while not endofsrc;
912   4           CALL FILLSOURCE;
913   4           if endofsrc and concat then
914   4               do; call setupeob;
916   5               ndest = nsbuf;
917   5               if nendcmd then return;
919   5               end;
920   4           ndest = nsbuf;
921   4           CALL WRITEDEST;
922   4           nsbuf = ndest;
923   4           if (endofsrc and insparc) then
924   4               call chkrandom;
925   4           end;

926   3       else do;
                  /* PERFORM THE ACTUAL COPY FUNCTION */
927   4           IF HEXT OR IGNOR THEN /* HEX FILE */
928   4               call hexrecord;
                  ELSE
929   4               DO WHILE NOT RD$EOF;
930   5               CALL PUTDEST(CHAR);
931   5               END;
932   4           if concat and nendcmd then
933   4               do; nsbuf = ndest;
935   5               return;
936   5               end;
937   4           end;

938   3       if dfile then
939   3           CALL CLOSEDEST;
940   3       END SIMPLECOPY;

941   2   MULTCOPY: PROCEDURE;
942   3       DECLARE (NEXTDIR, NDCNT, NCOPIED) ADDRESS;

943   3       PRNAME: PROCEDURE;
                  /* PRINT CURRENT FILE NAME */
944   4           DECLARE (I,C) BYTE;
945   4           CALL CRLF;
946   4               DO I = 1 TO FNSIZE;
947   5               IF (C := odest.fcb(I)) <> ' ' THEN
948   5                   DO; IF I = FEXT THEN CALL PRINTCHAR('.');
951   6                   CALL PRINTCHAR(C);
952   6                   END;
953   5               END;
954   4           END PRNAME;

955   3       archck: procedure byte;
                  /* check if archive bit is set in any extent of source file */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 956   4            if not archiv then
 957   4                return 1;
 958   4            call setsuser;
 959   4            source.fcb(12) = what;
 960   4            call search(.source);
 961   4                do while dcnt <> 255;
 962   5                call move(.buff+shl(dcnt and 11b,5)+1,.source.fcb(1),15);
 963   5                if not rol(source.fcb(11),1) then
 964   5                    return 1;
 965   5                call searchn;
 966   5                end;
 967   4            return 0;
 968   4            end archck;

                  /** $if mpm **/
                      /* initialize counters if not error retry */
 969   3            if eretry = 0 then NEXTDIR, NCOPIED = 0;
                  /** $else **/
                  /** $endif **/

 971   3            DO FOREVER;
                    /* FIND A MATCHING ENTRY */
 972   4            CALL SETSUSER; /* SOURCE USER */
 973   4            CALL SETDMA(.BUFF);
 974   4            searfcb(12) = 0;
 975   4            CALL SEARCH(.SEARFCB);
 976   4            NDCNT = 0;
 977   4                DO WHILE (DCNT <> 255) AND NDCNT < NEXTDIR;
 978   5                NDCNT = NDCNT + 1;
 979   5                CALL SEARCHN;
 980   5                END;
                    /* FILE CONTROL BLOCK IN BUFFER */
 981   4            IF DCNT = 255 THEN
 982   4                DO; IF NCOPIED = 0 THEN
 984   5                    call xerror(9,.searfcb);  /* file not found */
 985   5                if not kilds then
 986   5                    CALL CRLF;
 987   5                RETURN;
 988   5                END;
 989   4            NEXTDIR = NDCNT + 1;
                    /* GET THE FILE CONTROL BLOCK NAME TO DEST */
 990   4            CALL MOVE(.BUFF + SHL(DCNT AND 11B,5)+1,.odest.fcb(1),15);
 991   4            CALL MOVE(.odest.fcb(1),.SOURCE.FCB(1),15); /* FILL BOTH FCB'S */
 992   4            if archck then
 993   4                do; odest.fcb(12) = 0;
 995   5                source.fcb(12) = 0;
 996   5                IF RSYS OR NOT ROL(odest.fcb(10),1) THEN /* OK TO READ */
 997   5                    DO; if not kilds then    /* kill display option */
 999   6                        do; IF NCOPIED = 0 THEN
1001   7                            CALL PRINT(.('COPYING -$'));
1002   7                        dcnt = false;
1003   7                            do while ((dcnt <> 'Y') and (dcnt <> 'N'));
1004   8                            call prname;
1005   8                            if confrm then
1006   8                                do; call printx(.(' (Y/N)? $'));
1008   9                                dcnt = utran(rdchar);
1009   9                                end;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                                else
1010  8                                dcnt = 'Y';
1011  8                            end;
1012  7                        end;
1013  6                    ncopied = ncopied + 1;
1014  6                    made = false;  /* destination file not made */
1015  6                    if (dcnt = 'Y') or (Kilds) then
1016  6                        CALL SIMPLECOPY;
1017  6                    END;
1018  5                end;
1019  4            END;
1020  3        END MULTCOPY;

1021  2    CK$DISK: PROCEDURE;
           /* error if same user and same disk */
1022  3        IF (odest.user = source.user) and (odest.fcb(0) = source.fcb(0)) THEN
1023  3            CALL FORMERR;
1024  3        END CK$DISK;

1025  2    GNC: PROCEDURE BYTE;
1026  3        IF (CBP := CBP + 1) >= COMLEN THEN RETURN CR;
1028  3        RETURN UTRAN(COMBUFF(CBP));
1029  3        END GNC;

1030  2    DEBLANK: PROCEDURE;
1031  3            DO WHILE (CHAR := GNC) = ' ';
1032  4            END;
1033  3        END DEBLANK;

1034  2    CK$EOL: PROCEDURE;
1035  3        CALL DEBLANK;
1036  3        IF CHAR <> CR THEN CALL FORMERR;
1038  3        END CK$EOL;

1039  2    SCAN: PROCEDURE(FCBA);
1040  3        DECLARE FCBA ADDRESS,        /* ADDRESS OF FCB TO FILL */
           fcbs based fcba structure (    /* FCB STRUCTURE */
              fcb(frsize) byte,
           /** $if mpm **/
              pwnam(nsize) byte,
              pwmode byte,
           /** $endif **/
              user byte,
              type byte );
1041  3        DECLARE (I,K) BYTE;  /* TEMP COUNTERS */

           /* SCAN LOOKS FOR THE NEXT DELIMITER, DEVICE NAME, OR FILE NAME.
           THE VALUE OF CBP MUST BE 255 UPON ENTRY THE FIRST TIME */

1042  3        DELIMITER: PROCEDURE(C) BYTE;
1043  4            DECLARE (I,C) BYTE;
1044  4            DECLARE DEL(*) BYTE DATA
               (' =.:;,<>',CR,LA,LB,RB);
1045  4                DO I = 0 TO LAST(DEL);
1046  5                IF C = DEL(I) THEN RETURN TRUE;
1048  5                END;
1049  4            RETURN FALSE;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1050  4          END DELIMITER;

1051  3       PUTCHAR: PROCEDURE;
1052  4          FCBS.FCB(FLEN:=FLEN+1) = CHAR;
1053  4          IF CHAR = WHAT THEN AMBIG = TRUE; /* CONTAINS AMBIGUOUS REF */
1055  4          END PUTCHAR;

1056  3       FILLQ: PROCEDURE(LEN);
                   /* FILL CURRENT NAME OR TYPE WITH QUESTION MARKS */
1057  4          DECLARE LEN BYTE;
1058  4          CHAR = WHAT; /* QUESTION MARK */
1059  4             DO WHILE FLEN < LEN;
1060  5             CALL PUTCHAR;
1061  5             END;
1062  4          END FILLQ;

1063  3       SCANPAR: PROCEDURE;
1064  4          DECLARE (I,J) BYTE;
                   /* SCAN OPTIONAL PARAMETERS */
1065  4          CHAR = GNC; /* SCAN PAST BRACKET */
1066  4             DO WHILE NOT(CHAR = CR OR CHAR = RB);
1067  5             IF (I := CHAR - 'A') > 25 THEN /* NOT ALPHA */
1068  5                DO; IF CHAR = ' ' THEN
1070  6                   CHAR = GNC;
                      ELSE
1071  6                   call error(6);  /* BAD PARAMETER */
1072  6                END;
                   ELSE
1073  5                DO; /* SCAN PARAMETER VALUE */
1074  6                IF CHAR = 'S' OR CHAR = 'Q' THEN
1075  6                   DO; /* START OR QUIT COMMAND */
1076  7                   J = CBP + 1; /* START OF STRING */
1077  7                      DO WHILE NOT ((CHAR := GNC) = ENDFILE OR CHAR = CR);
1078  8                      END;
1079  7                    CHAR=GNC;
1080  7                   END;
1081  6                ELSE IF (J := (CHAR := GNC) - '0') > 9 THEN
1082  6                   J = 1;
                      ELSE
1083  6                   DO WHILE (K := (CHAR := GNC) - '0') <= 9;
1084  7                   J = J * 10 + K;
1085  7                   END;
1086  6                CONT(I) = J;
1087  6                IF I = 6 THEN /* SET SOURCE USER */
1088  6                   DO;
1089  7                   IF J > 15 THEN
1090  7                      call error(7);  /* INVALID USER NUMBER */
1091  7                   fcbs.user = J;
1092  7                   END;
1093  6                END;
1094  5             END;
1095  4          CHAR = GNC;
1096  4          END SCANPAR;


                /* scan procedure entry point */
```

CP/M plus ver 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
                    /* INITIALIZE FILE CONTROL BLOCK TO EMPTY */
1097   3            fcbs.type = ERR; CHAR = ' '; FLEN = 0;
                  /** $if mpm **/
1100   3                DO WHILE FLEN < (FRSIZE + NSIZE);
1101   4                IF FLEN = FNSIZE THEN CHAR = 0;
1103   4                ELSE IF FLEN = FRSIZE THEN CHAR = ' ';
                        call putchar;
1106   4                END;
1107   3            fcbs.pwnam(0) = 0;
1108   3            fcbs.pwmode = 1;
                  /** $else **/
                  /** $endif **/
1109   3            fcbs.fcb(0) = cdisk +1;    /* initialize to current disk */
1110   3            fcbs.user = cuser;    /* and current user */
                    /* CLEAR PARAMETERS */
1111   3                DO I = 0 TO 25; CONT(I) = 0;
1113   4                END;
1114   3            FEEDLEN,MATCHLEN,QUITLEN = 0;

                    /* DEBLANK COMMAND BUFFER */
1115   3            CALL DEBLANK;

                    /* CHECK PERIPHERALS AND DISK FILES */
                    /* SCAN NEXT NAME */
1116   3                DO FOREVER;
1117   4                FLEN = 0;
1118   4                    DO WHILE NOT DELIMITER(CHAR);
1119   5                    IF FLEN >= NSIZE THEN /* ERROR, FILE NAME TOO LONG */
1120   5                        RETURN;
1121   5                    IF CHAR = '*' THEN CALL FILLQ(NSIZE);
1123   5                    ELSE CALL PUTCHAR;
1124   5                    CHAR = GNC;
1125   5                    END;

                        /* CHECK FOR DISK NAME OR DEVICE NAME */
1126   4                IF CHAR = ':' THEN
1127   4                    DO; IF FLEN = 1 THEN
                            /* MAY BE DISK NAME A ... P */
1129   5                        DO;
1130   6                        IF (fcbs.fcb(0) := fcbs.fcb(1) - 'A' + 1) > 16 THEN
1131   6                            RETURN;  /* ERROR, INVALID DISK NAME */
1132   6                        CALL DEBLANK; /* MAY BE DISK NAME ONLY */
1133   6                        IF DELIMITER(CHAR) THEN
1134   6                            DO; IF CHAR = LB THEN
1136   7                                CALL SCANPAR;
1137   7                            CBP = CBP - 1;
1138   7                            fcbs.type = DISKNAME;
1139   7                            RETURN;
1140   7                            END;
1141   6                        END;
                            ELSE
                            /* MAY BE A THREE CHARACTER DEVICE NAME */
1142   5                        IF FLEN <> 3 THEN /* ERROR, CANNOT BE DEVICE NAME */
1143   5                            RETURN;
                            ELSE
                                /* LOOK FOR DEVICE NAME */
1144   5                            DO; DECLARE (I,J,K) BYTE, M LITERALLY '9',
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        IO(*) BYTE DATA
                        ('OUTPRNLSTAXO',
                          0,0,0,        /* fake area for file type */
                         'AUX',
                         'CONAXIINPNULEOF',0);

1146  6                 J = 255;
1147  6                    DO K = 0 TO M;
1148  7                    I = 0;
1149  7                       DO WHILE ((I:=I+1) <= 3) AND
                              IO(J+I) = fcbs.fcb(I);
1150  8                       END;
1151  7                    IF I = 4 THEN /* COMPLETE MATCH */
1152  7                       DO; fcbs.type = K;
                              /* SCAN PARAMETERS */
1154  8                       IF GNC = LB THEN CALL SCANPAR;
1156  8                       CBP = CBP - 1;
1157  8                       RETURN;
1158  8                       END;
1159  7                    J = J + 3;  /* OTHERWISE TRY NEXT DEVICE */
1160  7                    END;
1161  6                 RETURN;  /* ERROR, NO DEVICE NAME MATCH */
1162  6                 END;
1163  5              IF CHAR = LB THEN /* PARAMETERS FOLLOW */
1164  5                 CALL SCANPAR;
1165  5              END;
                    ELSE
                       /* CHAR IS NOT ':', SO FILE NAME IS SET. SCAN REMAINDER */
1166  4                 DO; IF FLEN = 0 THEN /* ERROR, NO PRIMARY NAME */
1168  5                    RETURN;
1169  5                 FLEN = NSIZE;
1170  5                 IF CHAR = '.' THEN /* SCAN FILE TYPE */
1171  5                    DO WHILE NOT DELIMITER(CHAR := GNC);
1172  6                    IF FLEN >= FNSIZE THEN
1173  6                       RETURN;  /* ERROR, TYPE FIELD TOO LONG */
1174  6                    IF CHAR = '*' THEN CALL FILLQ(FNSIZE);
1176  6                    ELSE CALL PUTCHAR;
1177  6                    END;
              /** $if mpm **/
1178  5                 FLEN = 0;
1179  5                 IF CHAR = ';' THEN  /* SCAN PASSWORD */
1180  5                    DO WHILE NOT DELIMITER(CHAR := GNC);
1181  6                    IF FLEN >= NSIZE THEN
1182  6                       /* ERROR, PW TOO LONG */ RETURN;
                          ELSE  /* SAVE PASSWORD */
1183  6                       FCBS.PWNAM(FLEN) = CHAR;
1184  6                       FLEN = FLEN + 1;
1185  6                    END;
              /** $endif **/
1186  5                 IF CHAR = LB THEN
1187  5                    CALL SCANPAR;
                          /* RESCAN DELIMITER NEXT TIME AROUND */
1188  5                 CBP = CBP - 1;
1189  5                 fcbs.type = FILE;
1190  5                 FCBS.FCB(32) = 0;
1191  5                 RETURN;
1192  5                 END;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1193  4            END;
1194  3        END SCAN;


            /* PLM (PIP) ENTRY POINT */
               /* BUFFER AT 80H CONTAINS REMAINDER OF LINE TYPED
               FOLLOWING THE COMMAND 'PIP' - IF ZERO THEN PROMPT TIL CR */

1195  2        if not retry then
1196  2            do; CALL MOVE(.BUFF,.COMLEN,80H);
1198  3            MULTCOM = (COMLEN = 0);

                  /* GET CURRENT CP/M VERSION */
1199  3            IF low(CVERSION) < VERSION THEN
1200  3                DO;
            /** $if cpm3 **/
1201  4                CALL PRINT(.('REQUIRES CP/M 3$'));
            /** $else **/
            /** $endif **/
1202  4                CALL BOOT;
1203  4                END;

1204  3            call mon1(45,255);  /* set return error mode */

            /** $if cpm3 **/
1205  3            call mon1(109,1);  /* set CP/M 3 control-C status mode */
            /** $endif **/

1206  3            if multcom then
1207  3                do;
            /** $if cpm3 **/
1208  4                call printx(.('CP/M 3 PIP VERSION 3.0$'));
            /** $else **/
            /** $endif **/
1209  4                call crlf;
1210  4                end;

1211  3            cuser,last$user = getuser;  /* GET CURRENT USER */
1212  3            cdisk = getdisk;            /* GET CURRENT DISK */
            /** $if mpm **/
1213  3            mseccnt = 1;
            /** $endif **/
1214  3            eretry = false; /* need to initialize here for first time */
1215  3            end;


               /* START HERE ON RESET EXIT FROM THE PROCEDURE 'ERROR' */
            /** $if mpm **/
1216  2        if eretry <> 0 then
1217  2            do; call multcopy;
1219  3            comlen = multcom;
1220  3            end;
            /** $endif **/
               /* MAIN PROCESSING LOOP.  PROCESS UNTIL CR ONLY */
1221  2            DO FOREVER;
1222  3            C1, C2, C3 = 0;     /* LINE COUNT = 000000 */
1223  3            CONCNT,COLUMN = 0;  /* PRINTER TABS */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
1224  3             ndest,nsbuf = 0;
1225  3             ambig = false;
1226  3             made = false;       /* destination file not made */
1227  3             opened = false;     /* source file not opened */
1228  3             concat = false;
1229  3             eretry = false;
1230  3             PUTNUM = TRUE;      /* ACTS LIKE LF OCCURRED ON ASCII FILE */
1231  3             dfile,sfile = true;
1232  3             nendcmd = true;
1233  3             LINENO = 254;       /* INCREMENTED TO 255 > PAGCNT */
                    /* READ FROM CONSOLE IF NOT A ONELINER */
1234  3             IF MULTCOM THEN
1235  3                 DO; CALL PRINTCHAR('*'); CALL RDCOM;
1238  4                 CALL CRLF;
1239  4                 END;
1240  3             CBP = 255;
1241  3             IF COMLEN = 0 THEN      /* character = <CR> */
1242  3                 do; call setcuser;  /* restore current user */
1244  4                 CALL BOOT;          /* normal exit from pip here */
1245  4                 end;

                    /* LOOK FOR SPECIAL CASES FIRST */

1246  3             CALL SCAN(.odest);
1247  3             if ambig then
1248  3                 call xerror(5,.odest);  /* invalid destination */
1249  3             call deblank;  /* check for equal sign or left arrow */
1250  3             if (char <> '=') and (char <> la) then call formerr;
1252  3             call scan(.source);

1253  3             IF odest.type = DISKNAME THEN
1254  3                 DO;
1255  4                 IF source.type <> file then call formerr;
1257  4                 CALL CK$EOL;
1258  4                 CALL CK$DISK;
1259  4                 odest.type = file;  /* set for character transfer */
                        /* MAY BE MULTI COPY */
1260  4                 IF AMBIG THEN  /* FORM IS A:=B:AFN */
1261  4                     DO;
1262  5                     CALL MOVE(.source.fcb(0),.searfcb(0),frsize);
1263  5                     CALL MULTCOPY;
1264  5                     END;
1265  4                 ELSE DO;  /* FORM IS A:=B:UFN */
1266  5                     CALL MOVE(.source.fcb(1),.odest.fcb(1),frsize - 1);
1267  5                     CALL SIMPLECOPY;
1268  5                     END;
1269  4                 END;

1270  3             else IF (odest.type = FILE) and (source.type = DISKNAME) THEN
1271  3                     DO;
1272  4                     CALL CK$EOL;
1273  4                     CALL CK$DISK;
1274  4                     source.type = file;  /* set for character transfer */
            /** $if mpm **/
1275  4                     call move(.odest.fcb(1),.source.fcb(1),(frsize+nsize));
            /** $else **/
            /** $endif **/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
1276  4                   CALL SIMPLECOPY;
1277  4                   END;

1278  3             else if (odest.type > cons) then
1279  3                 call error(3);  /* invalid destination */
1280  3             else do;
1281  4                 IF odest.type <> FILE THEN dfile = false;
              /** $if not mpm **/
                        /* no conditional attach list device */
              /** $else **/
              /** $endif **/
                        /* SCAN AND COPY UNTIL CR */
1283  4                   DO WHILE nendcmd;
1284  5                   sfile = true;
1285  5                   call deblank;
1286  5                   IF (CHAR <> ',' AND CHAR <> CR) THEN
1287  5                       call error(16);  /* invalid separator */
1288  5                   concat = concat or (nendcmd := (char = ','));
1289  5                   IF odest.type = PRNT THEN
1290  5                       DO; NUMB = 1;
1292  6                       IF TABS = 0 THEN TABS = 8;
1294  6                       IF PAGCNT = 0 THEN PAGCNT = 1;
1296  6                       END;
1297  5                   IF (source.type < file) or (source.type > eoft) or ambig THEN
1298  5                       call error(4);  /* invalid source */
1299  5                   IF source.type <> FILE THEN  /* NOT A SOURCE FILE */
1300  5                       sfile = false;
1301  5                   IF source.type = NULT THEN
                            /* SEND 40 NULLS TO OUTPUT DEVICE */
1302  5                       DO sfile = 0 TO 39; CALL PUTDEST(0);
1304  6                       END;
1305  5                   ELSE IF source.type = EOFT THEN
1306  5                       CALL PUTDEST(ENDFILE);
1307  5                   else call simplecopy;

1308  5                   CALL CK$STRINGS;
                          /* READ ENDFILE, GO TO NEXT SOURCE */

1309  5                   if nendcmd then call scan(.source);
1311  5                   END;
1312  4               end;

                  /* COMLEN SET TO 0 IF NOT PROCESSING MULTIPLE COMMANDS */
1313  3             COMLEN = MULTCOM;

1314  3             END; /* DO FOREVER */
1315  2         end plm;
1316  1         END;

              EOF
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

MODULE INFORMATION:

```
    CODE AREA SIZE     = 1FD4H   8148D
    VARIABLE AREA SIZE = 01D1H    465D
```

```
MAXIMUM STACK SIZE = 002CH      44D
1925 LINES READ
0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
ASM80 INPOUT.ASM DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        INPOUT   PAGE    1
INP:/OUT: Interface

  LOC  OBJ        SEQ        SOURCE STATEMENT

                    1 $title  ('INP:/OUT: Interface')
                    2         name inpout
                    3         cseg
                    4 ;
                    5 ;       CP/M 3 PIP Utility INP: / OUT: Interface module
                    6 ;       Code org'd at 080h
                    7 ;       July 5, 1982
                    8
                    9 public  inploc,outloc,inpd,outd
                   10
  0000              11        org     00h
                   12 inpd:
  0000 CD0800   C  13        call inploc
  0003 C9          14        ret
                   15
                   16 outd:
  0004 CD0B00   C  17        call outloc
  0007 C9          18        ret
                   19
                   20 inploc:
  0008 3E1A        21        mvi a,01Ah
  000A C9          22        ret
                   23
                   24 outloc:
  000B C9          25        ret
  000C 00          26        nop
  000D 00          27        nop
                   28
  007F             29        org     07fh
  007F 00          30        db      0
                   31 end

PUBLIC SYMBOLS
INPD   C 0000    INPLOC C 0008    OUTD   C 0004    OUTLOC C 000B

EXTERNAL SYMBOLS


USER SYMBOLS
INPD   C 0000    INPLOC C 0008    OUTD   C 0004    OUTLOC C 000B

ASSEMBLY COMPLETE,  NO ERRORS
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0000 PIP
0000 INPOUT
0180 INPD       0188 INPLOC     0184 OUTD       018B OUTLOC
0000 PIPMOD
2488 MEMORY     22B7 RETRY      05E8 PLM        0200 COPYRIGHT  22B8 COLUMN
22B9 LINENO     22BA FEEDBASE   22BB FEEDLEN    22BC MATCHLEN   22BD QUITLEN
22BE CDISK      22BF SBLEN      22C1 DBLEN      22C3 TBLEN      22C5 SBASE
2488 DBUFF      22C7 SOURCE     22F6 DEST       2325 ODEST      2354 FILSIZE
2317 DESTR      22E8 SOURCER    2319 DESTR2     22EA SOURCER2   2357 EXTSAVE
2358 NSBUF      235A BUFSIZE    235C MSECCNT    235D NSOURCE    235F NDEST
2361 FASTCOPY   2362 DBLBUF     2363 CONCAT     2364 AMBIG      2365 DFILE
2366 SFILE      2367 MADE       2368 OPENED     2369 ENDOFSRC   236A NENDCMD
236B INSPARC    236C SPARFIL    236D MULTCOh    236E PUTNUM     236F CONCNT
2370 CHAR       2371 FLEN       2372 F1         2373 F2         2374 F3
2375 F4         2376 RO         2377 SYS        2378 EXTEN      2379 ODCNT
237A ERETRY     237B DCNT       237C CBUFF      237C MAXLEN     237D COMLEN
237E COMBUFF    23FE CBP        23FF CUSER      2400 LASTUSER   2401 CONT
2401 ARCHIV     2403 CONFRM     2404 DELET      2405 ECHO       2406 FORMF
2407 GETU       2408 HEXT       2409 IGNOR      240B KILDS      240C LOWER
240E NUMB       240F OBJ        2410 PAGCNT     2411 QUITS      2412 RSYS
2413 STARTS     2414 TABS       2415 UPPER      2416 VERIF      2417 WRROF
241A ZEROP      241B ZEROSUP    241C C3         241D C2         241E C1
0884 RETCODES   241F A          0896 BOOT       089F RDCHAR     08A8 PRINTCHAR
2421 CHAR       08BA CRLF       08C5 PRINTX     2422 A          08D5 PRINT
2424 A          08E7 RDCOM      08F5 CVERSION   08FE SETDMA     2426 A
090E SETPW      2428 FCBA       0921 OPEN       242A FCBA       096A CLOSE
242C FCB        097F SEARCH     242E FCB        0994 SEARCHN    09A2 DELETE
2430 FCB        09BF DISKRD     2432 FCB        09D4 DISKWRITE  2434 FCB
09E9 MAKE       2436 FCBA       0A30 RENAME     2438 FCB        0A4D GETDISK
0A56 SETIND     243A FCB        0A6B GETUSER    0A74 SETUSER    243C USER
0A91 SETCUSER   0A99 SETDUSER   0AA1 SETSUSER   0AA9 RDRANDOM   243D FCB
0AC1 WRITERANDOM                243F FCB        0AD9 RETFSIZE   2441 FCB
0AE9 SETRANDOM  2443 FCB        0AF9 MULTSECT   2445 CNT        0B16 FLUSHBUF
0B1F CONATLST   0B28 MOVE       2446 S          2448 D          244A N
0220 ER00       022A ER01       0235 ER02       023C ER03       0250 ER04
025F ER05       026C ER06       027A ER07       028E ER08       029D ER09
02B1 ER10       02C0 ER11       02D0 ER12       02DF ER13       02F1 ER14
02FC ER15       0317 ER16       0329 ER17       033C ER18       035C ER19
0366 ER20       0370 ER21       037D ER22       0394 ERRMSG     03C2 SPER00
03D5 SPER01     03E3 SPER02     03FE SPER03     0417 SPER05     0432 SPER06
0446 SPER07     0454 SPER08     0465 SPER09     0472 SPECIALMSG 0486 EX00
0487 EX01       0496 EX02       049F EX03       04A8 EX04       04BC EX05
04CE EX07       04DF EX08       04EE EX10       04FD EXTMSG
0B5F ERRORCLEANUP               0BA5 ERROR      244B ERRTYPE    0BC3 XERROR
244C FUNCNO     244D FILEADR    244F TEMP       2450 I          2451 SDCNT
2452 SEXTEN     0513 MESSAGEINDEXTBL            0D48 FORMERR    0D4E CONBRK
0D6E MAXSIZE    0DAC SETUPDEST  0E57 SETUPSOURCE                2453 I
2454 J          0F09 WRITEDEST  2455 J          2456 DATAOK     2457 TDEST
2459 N          10C8 FILLSOURCE 11FB PUTDCHAR   245B B          1267 AXOCASE
12B9 PUTDESTC   245C B          245D I          1329 PRINT1     245E B
1350 PRINTDIG   245F D          136B NEWLINE    2460 ONE        13C5 PUTDEST
2461 B          2462 I          145B UTRAN      2463 B          1482 LTRAN
2464 B          14A9 GETSOURCEC 2465 B          2466 CONCHK     14C4 NOTSOURCE
1535 AXICASE    15C6 GETSOURCE  2467 CHAR       166B MATCH      2468 B
2469 C          16B0 RDEOF      16D5 HEXRECORD  246A H          246B HBUF
246C RL         246D CS         246E RT         246F ZEROREC    2470 LDA
17CC CKHEX      17FA RDHEX      1808 RDCS       181C RDADDR     1832 CKSTRINGS
184F CLOSEDEST  197E SIZEMEMORY 19E7 SETUPEOB   2472 I          1A41 SIMPLECOPY
2473 I          0524 OPTYPE     1B52 CHKRANDOM  1BE4 MULTCOPY   2474 NEXTDIR
2476 NDCNT      2478 NCOPIED    1D27 PRNAME     247A I          247B C
1D66 ARCHCK     1DB5 CKDISK     1DD5 GNC        1DF4 DEBLANK    1E03 CKEOL
1E12 SCAN       247C FCBA       247E I          247F K          2063 DELIMITER
2480 C          2481 I          053E DEL        2092 PUTCHAR    20B2 FILLQ
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
2482 LEN        20CC SCANPAR    2483 I          2484 J          2485 I
2486 J          2487 K          054A IO

0000 PIP#
0000 INPOUT#
0000 PIPMOD#
05E8 19         0884 33         088A 35         0891 36         0895 37
0896 38         0896 39         089E 40         089F 41         089F 42
08A8 43         08A8 44         08AC 46         08B9 47         08BA 48
08BA 49         08BF 50         08C4 51         08C5 52         08CB 54
08D4 55         08D5 56         08DB 58         08DE 59         08E6 60
08E7 61         08E7 62         08EC 63         08F4 64         08F5 65
08F5 66         08FE 67         08FE 68         0904 70         090D 71
090E 72         0914 75         0920 76         0921 77         0927 80
092F 81         093D 82         0956 84         095F 85         0964 86
0969 87         0969 88         096A 89         0970 91         097E 92
097F 93         0985 95         0993 96         0994 97         0994 98
09A1 99         09A2 100        09A8 102        09B0 103        09BE 104
09BF 105        09C5 107        09D3 108        09D4 109        09DA 111
09E8 112        09E9 113        09EF 116        09FC 117        0A0A 119
0A15 120        0A21 121        0A21 122        0A2F 123        0A30 124
0A36 126        0A3E 127        0A4C 128        0A4D 129        0A4D 130
0A56 131        0A56 132        0A5C 134        0A6A 135        0A6B 136
0A6B 137        0A74 138        0A74 139        0A78 141        0A82 142
0A90 143        0A91 144        0A91 145        0A98 146        0A99 147
0A99 148        0AA0 149        0AA1 150        0AA1 151        0AA8 152
0AA9 153        0AAF 155        0ABD 156        0AC1 157        0AC1 158
0AC7 160        0AD5 161        0AD9 162        0AD9 163        0ADF 165
0AE9 166        0AE9 167        0AEF 169        0AF8 170        0AF9 171
0AFD 173        0B07 174        0B15 175        0B16 176        0B16 177
0B1E 178        0B1F 179        0B1F 180        0B28 181        0B28 182
0B37 185        0B43 186        0B4D 187        0B54 188        0B5B 189
0B5E 190        0B5F 237        0B5F 238        0B64 239        0B69 240
0B70 242        0B73 243        0B79 244        0B7E 245        0B7E 246
0B85 248        0B88 249        0B8E 250        0B94 251        0B94 252
0B99 253        0B9E 254        0BA4 255        0BA5 256        0BA9 258
0BAC 259        0BBC 260        0BBF 261        0BC2 262        0BC3 263
0BCB 266        0BD1 267        0BD7 268        0BDA 269        0BF0 270
0BF8 271        0C00 272        0C17 273        0C42 275        0C47 276
0C59 277        0C5E 278        0C5E 279        0C66 281        0C6B 282
0C7B 283        0C7B 284        0C81 285        0C8D 287        0C98 288
0C9D 289        0CAB 290        0CC0 292        0CC8 293        0CCD 294
0CD4 295        0CD4 296        0CDB 297        0CDB 298        0CDE 299
0D0E 300        0D17 301        0D3E 302        0D44 303        0D47 304
0D48 305        0D48 306        0D4D 307        0D4E 308        0D4E 309
0D5B 310        0D68 311        0D6D 312        0D6E 313        0D6E 314
0D78 315        0D7B 316        0D85 318        0D8F 319        0D92 320
0D9C 322        0DA6 323        0DA9 324        0DA9 325        0DA9 326
0DAC 327        0DAC 328        0DAC 329        0DAF 330        0DBB 331
0DC7 332        0DCF 333        0DD5 334        0DDB 335        0DE0 336
0DE9 337        0DF3 338        0DFB 339        0E01 340        0E19 341
0E21 342        0E27 343        0E2F 344        0E39 345        0E44 346
0E4C 347        0E51 348        0E56 349        0E57 350        0E57 352
0E5A 353        0E62 354        0E68 355        0E70 356        0E75 357
0E85 358        0E8A 359        0E92 360        0E9C 361        0EA7 362
0EAF 363        0EB7 364        0EBF 365        0EC7 366        0ECF 367
0ED7 368        0EDF 369        0EE8 370        0EF4 371        0EF9 372
0F02 373        0F08 374        0F09 375        0F09 377        0F10 378
0F13 379        0F28 380        0F29 381        0F2F 382        0F32 383
0F3E 385        0F43 386        0F4F 387        0F5A 388        0F62 389
0F65 390        0F6B 391        0F72 393        0F78 394        0F7D 395
0F80 397        0F86 398        0F8B 399        0F8B 400        0F9C 401
0FB5 403        0FC1 404        0FCE 405        0FCE 406        0FDA 407
0FE0 408        0FE8 409        0FF0 410        0FFB 411        0FFE 412
1005 414        1008 415        100E 416        1013 417        1019 418
1025 419        1033 420        1043 421        1047 422        104C 423
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

105A 424 107A 425 107E 426 1081 427 108B 428
1092 429 109A 430 109D 431 10A3 432 10A3 433
10A9 434 10C7 435 10C8 436 10C8 437 10CB 438
10D2 440 10D8 441 10DD 442 10E0 444 10E6 445
10EB 446 10EB 447 10EE 448 10F4 449 1105 450
11[illegible] 452 1130 453 113D 454 113D 455 114A 456
1150 457 1156 458 115E 460 1166 461 116E 462
1175 463 1187 464 11A1 465 11A6 466 11AC 467
11B7 469 11C3 470 11CB 471 11CE 473 11D4 474
11D9 475 11D9 476 11DE 477 11E8 478 11E9 479
11EC 480 11F7 481 11FA 482 11FB 483 11FF 485
1207 487 120B 488 1214 490 121E 491 121F 492
121F 493 121F 494 1226 495 1231 496 1241 497
124B 498 1259 499 1267 500 1275 501 1275 502
12[illegible] 503 1284 504 128F 505 1296 506 1299 507
12[illegible] 508 12AA 509 12B8 510 12B9 511 12BD 513
12C5 514 12CF 515 12D7 516 12E1 518 12E7 519
12F1 520 12FB 521 12FE 522 1306 523 130F 524
1313 525 1318 526 131B 527 131B 528 1323 529
1328 530 1329 531 132D 533 133E 534 1346 535
134F 536 1350 537 1354 539 1361 540 136A 541
13[illegible] 542 136B 544 1370 545 137B 546 1383 547
138C 548 1395 549 139C 550 13A3 551 13AA 552
13B2 554 13B7 555 13BC 556 13BF 557 13C4 558
13C5 559 13C9 561 13D0 563 13D8 564 13D9 565
13D9 566 13E0 568 13F8 570 1403 572 140B 573
1410 574 141E 576 1423 577 1428 578 142B 579
1428 580 1431 581 1434 582 1439 583 1439 584
1439 585 1441 586 1446 587 144D 588 1455 589
145A 590 145B 591 145F 593 1476 594 147E 595
1482 596 1482 597 1486 599 149D 600 14A5 601
14A9 602 14A9 603 14A9 605 14AE 606 14BE 607
14C1 608 14C4 609 14CC 610 14CF 611 14CF 612
14DB 614 14E7 615 14F0 616 14FD 618 1500 619
1506 620 1506 621 1509 622 1509 623 1515 624
151C 625 151F 626 1522 627 1522 628 1527 629
1532 630 1535 631 1545 632 154E 633 1560 634
1567 636 156E 637 1580 638 158B 639 1592 641
1595 642 1595 643 1595 644 159C 645 15A4 646
15AB 647 15B3 648 15BA 649 15C2 650 15C6 651
15C6 652 166B 654 166F 656 1687 658 1694 659
1697 660 1697 661 16A1 662 16A8 663 16AD 664
16B0 665 15C6 666 15CF 668 15DB 669 15DE 670
15E1 671 15E1 672 15E1 673 15EA 675 15EE 676
15FA 677 15FE 678 15FF 679 15FF 680 160A 681
160D 682 1616 684 1621 686 1627 687 162C 688
1633 689 1638 690 1638 691 163B 692 1644 694
164F 696 1654 697 1659 698 165C 699 165C 700
1660 701 1663 702 1667 703 166A 704 16B0 705
16B0 706 16B6 707 16BD 708 16CC 709 16D5 710
16D5 711 17CC 713 17CC 714 17D8 715 17DE 716
17EA 717 17F2 718 17FA 719 17FA 720 17FA 721
1804 722 1808 723 1808 724 1808 725 181C 726
181C 727 181C 728 1832 729 16D5 730 16DA 731
16E0 732 16E0 733 16E8 734 16F0 736 16F7 737
16F8 738 1700 739 1700 740 1707 741 170D 742
1710 743 1716 744 1720 745 1726 746 172C 747
1734 748 173C 749 1744 750 1749 751 1754 752
176C 753 1772 754 1778 756 177D 757 1784 758
178B 759 1791 760 1797 761 179D 762 17A5 763
17A9 764 17AF 765 17B2 766 17BA 767 17C2 768
17C8 769 17C8 770 17CB 771 1832 772 1832 773
183B 774 1840 775 1849 776 184E 777 184F 778
184F 779 185A 780 185F 781 1862 782 1865 783
186B 784 186B 785 1871 786 1879 787 1881 788
1889 790 1891 792 1898 794 18B0 795 18B6 796

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

18C0 797 18C3 798 18CB 800 18D1 801 18D4 802
18DA 803 18DB 804 18DB 805 18DE 806 18DE 807
18DE 808 18E6 809 18EE 810 18F4 811 18FA 812
18FA 813 1906 814 190C 815 1918 816 1922 817
192C 818 1936 819 193E 820 1948 821 1952 822
195A 823 1960 824 1967 82o 196A 827 1972 828
1977 829 197D 830 197D 831 197E 832 197E 833
1985 835 198B 836 19A3 837 19A6 839 19C6 840
19D1 841 19DD 842 19E0 843 19E6 844 19E6 845
19E7 846 19E7 848 19FF 850 1A0A 851 1A18 852
1A2C 854 1A38 855 1A39 856 1A39 857 1A40 858
1A40 859 1A41 860 1B52 864 1B52 865 1B55 866
1B5B 867 1B60 868 1B66 869 1B66 870 1B7F 872
1B85 873 1B8B 874 1B90 875 1B91 876 1B91 877
1B99 879 1BA9 880 1BAD 88' 1BB0 882 1BB8 884
1BD1 885 1BD5 886 1BD8 88/ 1BE0 888 1BE3 889
1A41 890 1A4B 891 1A50 892 1A55 893 1A5A 894
1A5D 895 1A6B 896 1A7A 898 1A89 899 1A91 900
1AA0 902 1AAA 903 1AAF 904 1AAF 905 1AAF 906
1AB6 907 1AB9 908 1AC0 909 1AC3 910 1ACA 911
1AD1 912 1AD4 913 1ADF 915 1AE2 916 1AE8 917
1AEF 918 1AF0 919 1AF0 920 1AF6 921 1AF9 922
1AFF 923 1B0A 924 1B0D 925 1B13 927 1B1E 928
1B24 929 1B2B 930 1B32 931 1B35 932 1B40 934
1B46 935 1B47 936 1B47 937 1B47 938 1B4E 939
1B51 940 1BE4 941 1D27 943 1D27 945 1D2A 946
1D38 947 1D4A 949 1D52 950 1D57 951 1D5E 952
1D5E 953 1D65 954 1D66 955 1D66 956 1D6D 957
1D70 958 1D73 959 1D78 960 1D7E 961 1D86 962
1DA1 963 1DA9 964 1DAC 965 1DAF 966 1DB2 967
1DB5 968 1BE4 969 1BEC 970 1BF5 971 1BF5 972
1BF8 973 1BFE 974 1C03 975 1C09 976 1C0F 977
1C29 978 1C30 979 1C33 980 1C36 981 1C3E 983
1C4A 984 1C52 985 1C59 986 1C5C 987 1C5D 988
1C5D 989 1C64 990 1C7F 991 1C8B 992 1C92 994
1C97 995 1C9C 996 1CA9 998 1CB0 1000 1CBC 1001
1CC2 1002 1CC7 1003 1CDF 1004 1CE2 1005 1CE9 1007
1CEF 1008 1CF9 1009 1CFC 1010 1D01 1011 1D04 1012
1D04 1013 1D0B 1014 1D10 1015 1D20 1016 1D23 1017
1D23 1018 1D23 1019 1D26 1020 1DB5 1021 1DB5 1022
1DD1 1023 1DD4 1024 1DD5 1025 1DD5 1026 1DE3 1027
1DE6 1028 1DF4 1029 1DF4 1030 1DF4 1031 1DFF 1032
1E02 1033 1E03 1034 1E03 1035 1E06 1036 1E0E 1037
1E11 1038 1E12 1039 2063 1042 2067 1045 2075 1046
2085 1047 2088 1048 208F 1049 2092 1050 2092 1051
2092 1052 20A4 1053 20AC 1054 20B1 1055 20B2 1056
20B6 1058 20BB 1059 20C5 1060 20C8 1061 20CB 1062
20CC 1063 20CC 1065 20D2 1066 20EA 1067 20F9 1069
2101 1070 210A 1071 210F 1072 2112 1074 212A 1076
2131 1077 214C 1078 214F 1079 2155 1080 2158 1081
216A 1082 2172 1083 2184 1084 2199 1085 219C 1086
21A9 1087 21B1 1089 21BA 1090 21BF 1091 21CA 1092
21CA 1093 21CA 1094 21CD 1095 21D3 1096 1E18 1097
1E21 1098 1E26 1099 1E29 1100 1E31 1101 1E39 1102
1E41 1103 1E49 1104 1E4E 1105 1E51 1106 1E54 1107
1E5D 1108 1E66 1109 1E6E 1110 1E77 1111 1E85 1112
1E90 1113 1E97 1114 1EA2 1115 1EA5 1116 1EA5 1117
1EAA 1118 1EB5 1119 1EBD 1120 1EBE 1121 1EC6 1122
1ECE 1123 1ED1 1124 1ED7 1125 1EDA 1126 1EE2 1128
1EEA 1130 1EFD 1131 1EFE 1132 1F01 1133 1F0C 1135
1F14 1136 1F17 1137 1F1B 1138 1F24 1139 1F25 1140
1F25 1141 1F28 1142 1F30 1143 1F31 1146 1F36 1147
1F42 1148 1F47 1149 1F7B 1150 1F7E 1151 1F86 1153
1F91 1154 1F99 1155 1F9C 1156 1FA0 1157 1FA1 1158
1FA1 1159 1FA9 1160 1FB0 1161 1FB1 1162 1FB1 1163
1FB9 1164 1FBC 1165 1FBF 1167 1FC7 1168 1FC8 1169

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

1FCD 1170	1FD4 1171	1FE2 1172	1FEA 1173	1FEB 1174
1FF3 1175	1FFB 1176	1FFE 1177	2001 1178	2006 1179
200D 1180	201B 1181	2023 1182	2024 1183	2036 1184
203A 1185	203D 1186	2045 1187	2048 1188	204C 1189
2055 1190	205E 1191	205F 1192	205F 1193	2062 1194
05E8 1195	05EF 1197	05FB 1198	0606 1199	060F 1201
0615 1202	0618 1203	0618 1204	0620 1205	0628 1206
062F 1208	0635 1209	0638 1210	0638 1211	0641 1212
0647 1213	064C 1214	0651 1215	0651 1216	0659 1218
065C 1219	0662 1220	0662 1221	0662 1222	066D 1223
0677 1224	0680 1225	0684 1226	0687 1227	068A 1228
068D 1229	0690 1230	0695 1231	069D 1232	06A2 1233
06A7 1234	06AE 1236	06B3 1237	06B6 1238	06B9 1239
06B9 1240	06BE 1241	06C6 1243	06C9 1244	06CC 1245
06CC 1246	06D2 1247	06D9 1248	06E1 1249	06E4 1250
06FC 1251	06FF 1252	0705 1253	070D 1255	0715 1256
0718 1257	071B 1258	071E 1259	0723 1260	072A 1262
0736 1263	0739 1264	073C 1266	0748 1267	074B 1268
074B 1269	074E 1270	0766 1272	0769 1273	076C 1274
0771 1275	077D 1276	0780 1277	0783 1278	078C 1279
0794 1281	079C 1282	07A1 1283	07A8 1284	07AD 1285
07B0 1286	07C8 1287	07CD 1288	07DD 1289	07E5 1291
07EA 1292	07F2 1293	07F7 1294	07FF 1295	0804 1296
0804 1297	081D 1298	0822 1299	082A 1300	082F 1301
0837 1302	0845 1303	084A 1304	0854 1305	085C 1306
0864 1307	0867 1308	086A 1309	0871 1310	0877 1311
087A 1312	087A 1313	0880 1314	0883 1315	

CP/M plus Ver 3.0

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. # CR3-135